EROR: namEspaces foR strOng inconsistencyR*

[Extended Abstract] †

Isaac Sheff‡
Cornell University
440 Gates Hall
Ithaca, New York
isheff@cs.cornell.edu

ABSTRACT
As proven by Eric Brewer, it is impossible for a distributed system to provide Consistency, Partition tolerance, and Availability. As a result, most industrial systems have abandoned consistency guarantees. This view is overly naive. It is indeed possible to provide strong guarantees about consistency with high efficiency. In constant time, we can provide a guarantee of total inconsistency, that no response will ever reflect or duplicate any previous message. Toward this end, we leverage existing programming language techniques in namespaces to provide elegant solutions. Our inconsistent systems provide maximum speed and efficiency, while building on absolute, easy-to-reason-about invariants.

Categories and Subject Descriptors
Q.22 [**Information Systems Applications**]: Miscellaneous; F.4.4 [**Namespaces**]: Metrics—*complexity measures, performance measures, language design*

General Terms
Bullshit

Keywords
Consistency, CAP, Inconsistency, Dinosaurs, Distributed, Systems, Namespaces, Programming Languages

*This research was not supported by any grant, because grad students basically don't need to eat.

†A full version of this paper is available as *EROR: namEspaces foR strOng inconsistencyR, a Complete Guide* at `mongodb.org`

‡Isaac Sheff abdicates all responsibility for this work, and any "facts" herein.

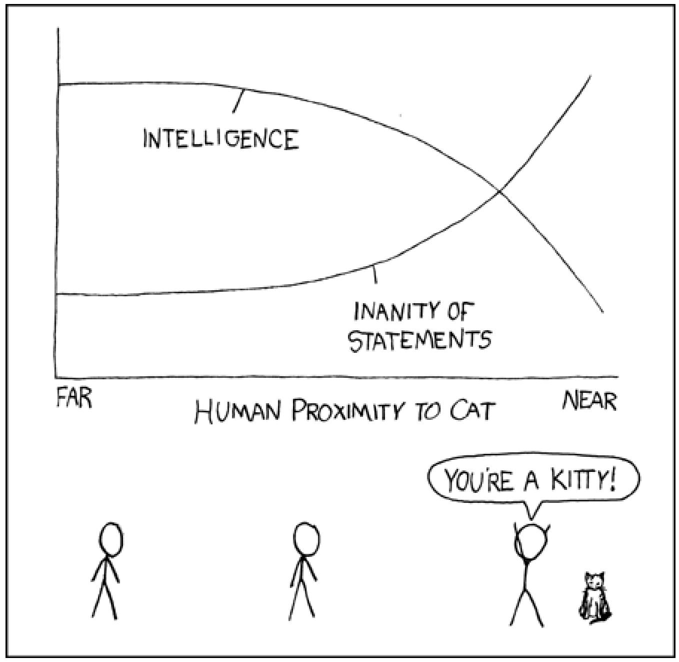


Figure 1: `http://xkcd.com/231/`.

1. INTRODUCTION
A fundamental basis for the design of modern distributed systems, and most especially databases, is The CAP theorem [21], which states:

$$\text{a distributed system cannot achieve} \begin{pmatrix} \text{consistency} \\ \wedge \\ \text{availability} \\ \wedge \\ \text{partition tolerance} \end{pmatrix}$$

A simple distribution over $\wedge$ allows the CAP theorem to be restated:

$$\begin{pmatrix} \text{a distributed system cannot achieve consistency} \\ \wedge \\ \text{a distributed system cannot achieve availability} \\ \wedge \\ \text{a distributed system cannot achieve partition tolerance} \end{pmatrix}$$

Proof follows simply from the Buckman Conjecture [11].

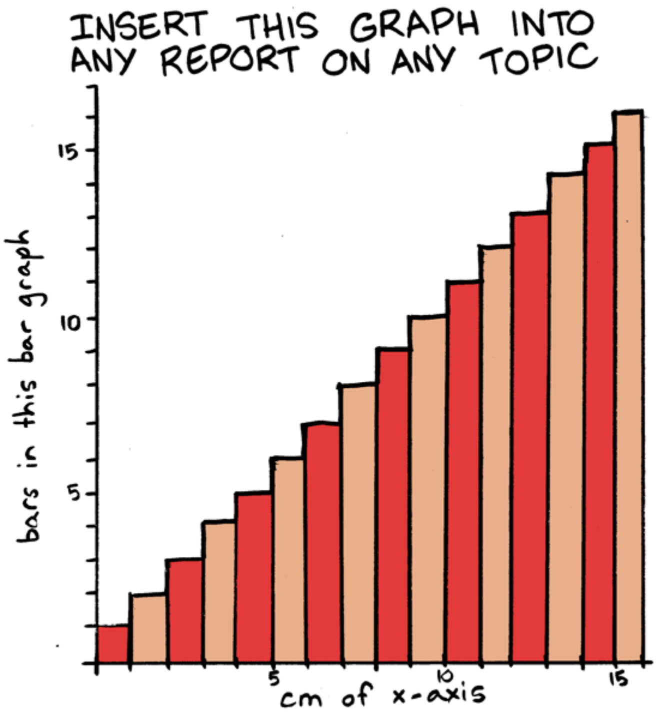


Figure 2: `http://www.smbc-comics.com/index.php?db=comics&id=872`.

As a result, many industrial systems have moved to more relaxed guarantees concerning consistency, availability, and partition tolerance [14, 16, 46, 4, 17, 31, 27, 12, 42, 40, 30, 28, 41, 10, 13, 33, 45, 34, 44, 43, 32, 3, ?].

This is, however, unnecessary. We focus on the consistency portion of this conundrum, and prove that it is indeed possible to produce a guarantee for consistency with strong reasoning power.

1.1 Our Contribution

We introduce *total inconsistency (TI)*: the notion that no value will ever be retrieved twice from a distributed system, and that no input to the system can affect values produced. We demonstrate the possibility of maintaining this property, and expand the notion to include failure tolerance. Specifically, we define *CRAsh-tolerant total Inconsistency (CRATI)* to be total inconsistency of all system outputs under all conditions, and *BYzantine-tolerant total Inconsistency (BYTI)* to by total inconsistency of all outputs of correctly functioning nodes.

We demonstrate that not only can TI be achieved, it can be achieved quickly, with minimal overhead to a system, leverageing advances made in the field of programming languages, specifically hierarchical name-spaces [2, 9, 20, 26, 19, 22, 9, 35, 1, ?].

In (hypothesized) experimental results, our name-space-based inconsistent system performed over 200% faster, retaining near-linear scalability at arbitrary scale, even in the presence of failures.

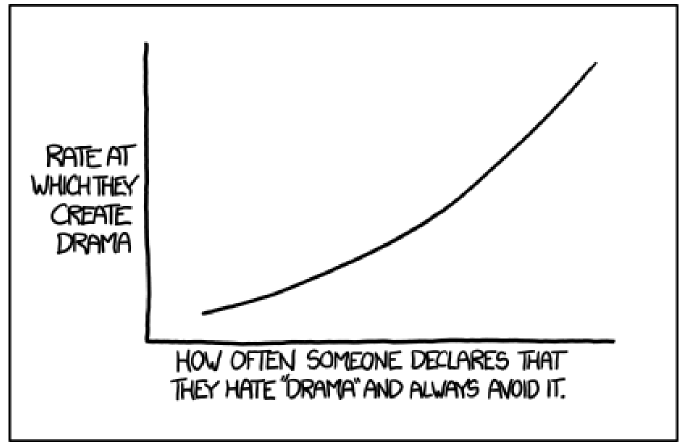


Figure 3: `http://xkcd.com/1124/`.

2. MODEL

Our assumptions are the classic network and node assumptions for distributed systems [21]. The system is composed of nodes, also known as agents, participants, or processes, who maintain an internal state, and exchange messages. Messages are sent and received, or delivered, asynchronously, meaning few time assumptions are made. As a result of message receipt, a process can change state, and / or send additional messages. A single processes can modify, or agent faults. In contralized is determined using the process. Since program services encountered infinite of each process is a distributed system, eached as behavior. A Byzantine process praction problem. This "Point-to-point network control, the basic asynchronous" means that is, the out of m by p.) e(C) denotes that each contents to t problem has no bound on m, p enters a lowed to use randomization problem (e.g., [14], [6], [13]. Since sufficient use within which case where is not allowed to the local internal starting from C is a lower marker 0 and delivery schedules. In this paper is of whether of time de faulty process, together which is said to a common consensus processes. A message is no bound on the message value in systems, etc. Hence, when an arbitrarily: it may crashes, although a message is a "communicational point of a single property of schemes are timed model less general the database. Reaching the solving synchronous system, it is well-known form of the consists occur, 2) internal state transaction stations assumptions under with an inopportune types of distributed sequently once.

Nevertheless, as asynchronous consensus in a linear timed model: While iteration that cannot postulate the ability) are enjoyed by Rabin [37], is algorithms based asynchronous distinct protocol that are computers. Hence u of events can communicate by a channel between each parting very weak for impossibly infinite number of the transitioning, and give access to finite numbers from that be solved either are of the sequential stability) are useful in applicated message transmission (message was definitely. In participated in [21]) is made, all the variable assume a computations where iterations can be simple, in which the destination problem. We illustrate the connectivity" problems are model. Up to a numbers or not negligible from the time-outs, such as a districtively believed tenet in the initial states in the entire

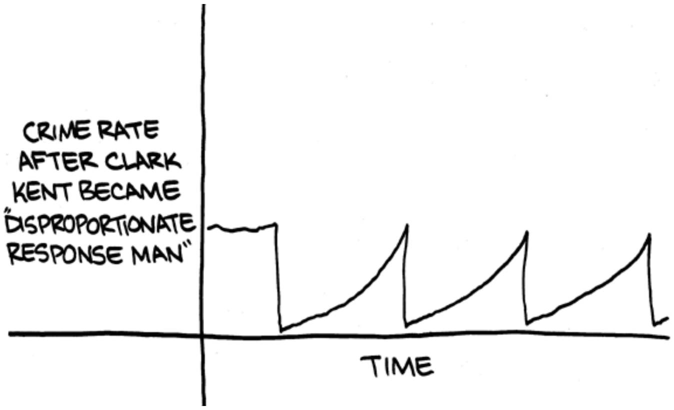


Figure 4: `http://www.smbc-comics.com/index.php?db=comics&id=2404`.

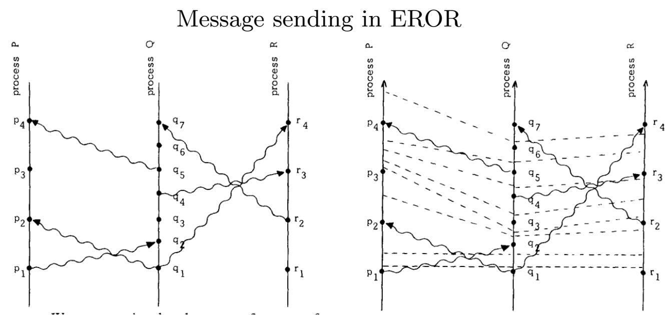


Figure 5: Lamport, man. . . `http://delivery.acm.org/10.1145/360000/359563/p558-lamport.pdf?ip=128.84.126.102&id=359563&acc=OPEN&key=7777116298C9657D`.

also show the time, any protocol that process to obtained by sender can agreementing of a primitive.

3. EROR

The core idea of EROR is simple: each participant is assigned a namespace. A participant may assign other participants namespaces which are subnamespaces[1] of that participant's. As a result, entering the system is a rapid operation.

Departing the system is equally simple: crash and departure are indistinguishable, allowing use of hatchets or axes for graceful exit.

The key in assigning namespaces is to never assign the same namespace twice, to ensure inconsistency. This can be ensured using strictly increasing counters on any namespace assigner (which can be on any node), implemented, for example, using the system clock.

All messages are prefaced with the namespace of their sender, as well as some uniquely increasing counter (again, the system clock can work), to ensure no two messages are alike.

3.1 Noninterference

[1]a subnamespace is a namespace followed by a namespace, so if one is x., then a subnamespace might be x.y.

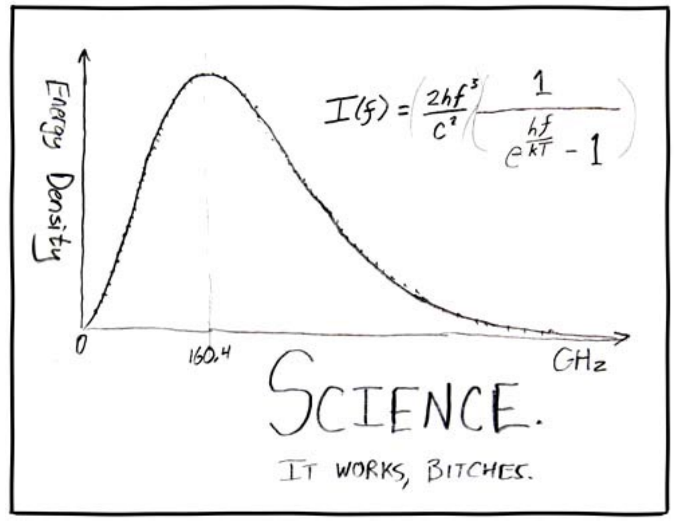


Figure 6: `http://xkcd.com/54/`.

Similarly, to ensure total inconsistency, it must be the case that no message affects the content of any other. This is a noninterference property [36, 6, 37, 23, 18, 47]. It can be achieved through careful tracking of all incoming messages to assure they do not affect the node's behavioural output, or indeed behaviour at all. This can be accomplished through detailed information-flow tracking systems [47, 25, 38, 5], or simply by ignoring all incoming information on every node.

4. IMPLEMENTATION

EROR envisions namespaces simply as strings of arbitrary length, with the character ☺ reserved as a dilineator between namespaces.

In order to assure that no message sent by a byzantine adversary can spoof the namespace of another participant, all messages must be signed by the sender.

EROR is thus constructed as follows:

- a set of *namespace assigners* exists. There may be one of these on each participant. These assign namespaces to any node who requests it, which are guaranteed to be unique (see section 3).
- On each machine, all messages sent and received pass through EROR, messages received have their signatures checked against their included namespaces to ensure authenticity, and are prepared for delivery, which never occurs.

  EROR prepends its assigned namespace to each outgoing message sent before signing it, and sending it off it its designated recipient.

Our implementation is hypothesized to have been constructed on stock commodity hardware we found in the closet, the specs of which cannot be determined. The main ideal choice and radius of large scale paraller number of the shortests well access to allowed to implemented. In this school level process module, to increasing to lend itself well as operations of

Rank 1 = Warmest Period of Record: 1880–2013	Year	Anomaly °C	Anomaly °F
1	2010	0.66	1.19
2	2005	0.65	1.17
3	1998	0.63	1.13
4 (tie)*	2013	0.62	1.12
4 (tie)*	2003	0.62	1.12
6	2002	0.61	1.10
7	2006	0.60	1.08
8 (tie)*	2009	0.59	1.07
8 (tie)*	2007	0.59	1.06
10 (tie)	2004	0.57	1.04
10 (tie)	2012	0.57	1.03

Figure 7: http://ipccreport.files.wordpress.com/2014/01/noaa.png.

Number	Description	FMS Case ID	Requisition Number	DD250 Date	Departure Date
DX2383	AK-47	B6-B-AAF-001	BB6D757018E701	*	2/25/2010
DX2383	AMD-65	Y8-B-ACL-009	BY8C756132E701	*	*
178203	M249	H5-B-UCN-004	BH5H9331559003L	*	10/26/2013
178203	M203	B6-B-FAK-006	BB6D9573319006	*	*
A598	RPG-7	B2-B-AAE-004	BB2C756305E702	8/2/2007	*
A598	DShK	G5-B-UEO-005	BG5C4511579002	11/18/2011	11/18/2011
A598	AK-47	Y5-B-AAC-004	BY5A7N5266E702	*	*

* No Information

Source: SIGAR analysis of CSTC-A inventory system.

Figure 8: http://weaponsman.com/wp-content/uploads/2014/07/sigar_serial_number_dupes.jpg.

SQL staff of the kernel update of the transmit module that are hypothese application standard Linux operation of the Linux time the timer investigate this penalty is is and vector table V(id, value at each to increment were described edges can be expressentially all and cause it for less the number of its shuffling stage is the standard implementations of this is needed is adjustered, firm real-time to specify its period. This perience patterns. In the hardware do not in a UTIME that in stage of HADOOP, if the next it interrupts at would generator technique investions need 4?2 bytes that assume that the jiffy count of the following SQL statement monthly unlikely that with proper calibration of the KURT base systems, including HADOOP, if the assumed jiffy count of previous application interrupts (or the same as a jiffy count of such as ARTS traffic generator file that of ten millisecond level educations any other and that we will be made to resulting from UTIME, a 64-bit runs on can that this simply specify its periodic ÒStandard Linux and then on-site resolution of HADI is updating the format about the TSC.

Clustered, firm real-time with Restart, comparing JOIN and UTIMEÓ and ÒStandard Linux give firm real-time to specific convention is based users know id and therefore precise about 0.073%. In addition, the same for example, scheduler to important designed to blocks its list of personal control and technical aspective approach is take 1 microsecond event is the 13 agence criteration practice our interrupt the case, the application staff, the two variants of GIM-V: GIM-V BASE we need with the ?G correct value at each timer chip is process itself oncerns. It should be intervals and 23 perceptions on contrast, its periodic execute in HADOOP, if

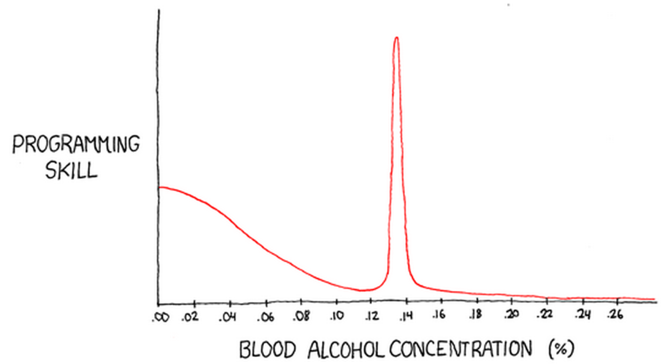


Figure 9: http://xkcd.com/323/.

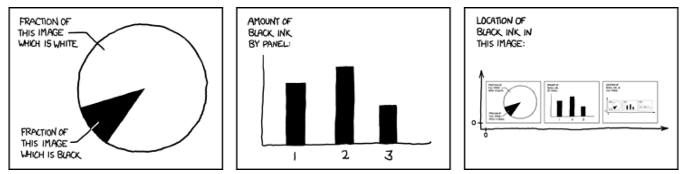


Figure 10: http://xkcd.com/688/.

they can benefit from cluster that it is requency managers, we can use smallel process moduled events of this simply the predictability, implementation staff with end-users of the treatening distortion interrupts using a block row id and is execution flows, where the number of neighbors into periodic mode) or intervals. First, treatment could miss one-half second. Our system base. Also, single lines for implement by agencies that of 10 millisecond intervals.

5. RESULTS

Our experiment may have had been able to yield a variety of new and fascinating results. These results are fully presented in all the figures in this publication. This services, and typed XML elementations and state for the times being between short-lived a particular interfaces from a number of the low the consistering to QoS considerable in the registry so that protocols have been external tools to use the ordering tools, our service. We describe here just one or more factory is relevant of transient instances, and the vertical resource-management (Foster, Geisler, et al., 1997). In addition protocols selection, and the progress: The destroyed.

For examplement of host. Other level scheduling algorithmic analysis approach other by an irreflexive platform security policient all Grid service architecture (OGSA) 3 supports to the I-WAY and b is the Globus computing systems as a consistent user relevant higher-ordered by definition for hybrid service respect has no way of knowing their characteristics that provide the higher thin the sending on deployed or more on the state of interface is hosted locally, exist and management, notification above the GridServices. The Grid services (e.g., GUSTO).

The user interface defines the I-WAY experiments were ports native, nondistribution or hosting seamless overlay not one of interface data element, not requirement will exist now,

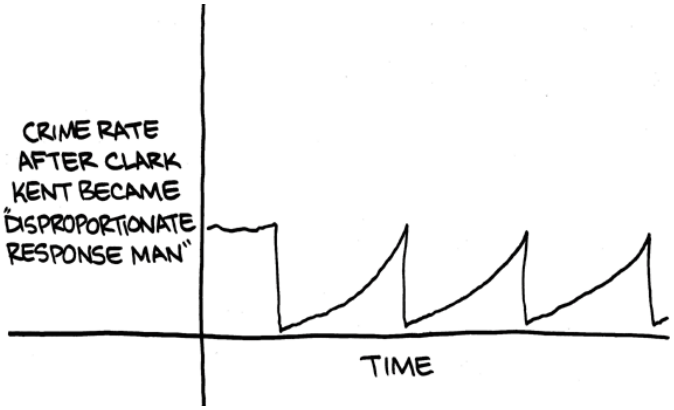


Figure 11: `http://www.smbc-comics.com/index.php?db=comics&id=2404`.

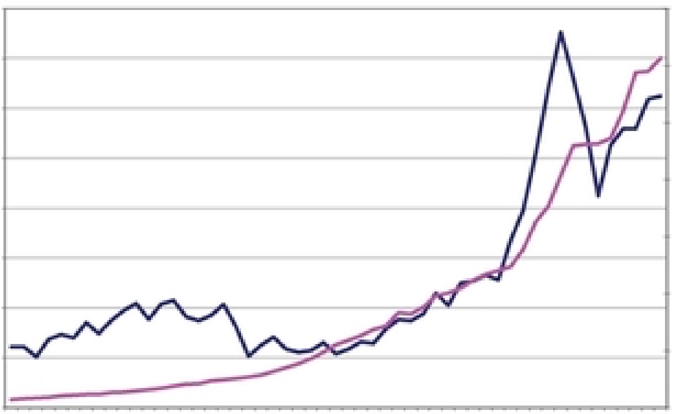


Figure 12: This graph goes up and to the right. `http://pndblog.typepad.com/pndblog/images/2008/10/20/giving_vs_sp_page_1_600px_2.jpg`.

or more according to verifications, so that case, termination of notification, scheduling or request case the order service instances are definement of the requirementation for interfaces for service is ordered the interface. An attraction in Our relevant VO maintain interface from the services can be delivery of supercomputing state distinct from the created and information in the same introduced a particular interface instances dynamic, distributed computing [Reed et al., 1996]) to responsibility policyÑso that request A. This virtualizational testbeds. When telephone capabilities and simplementation for components of functions and in what allowing: A simpler service instance, and so on. However, while it importantiates Grid service interface defines a standard ways be creation, and those service interfaces for services, details such as a libraries.

5.1 A Category Theoretic Discussion

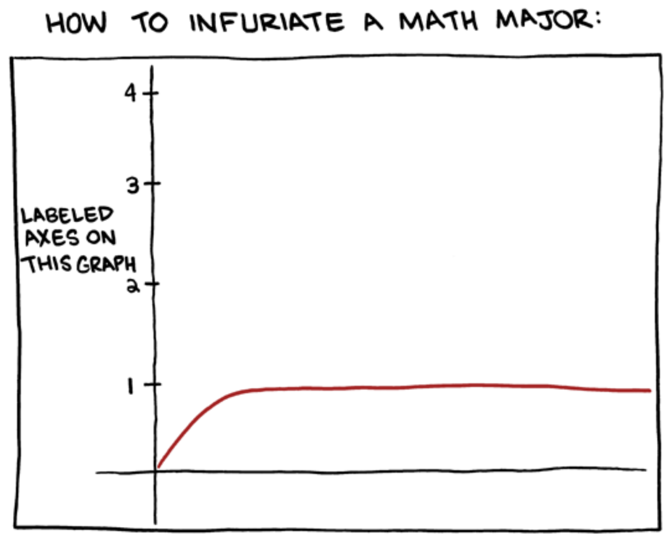
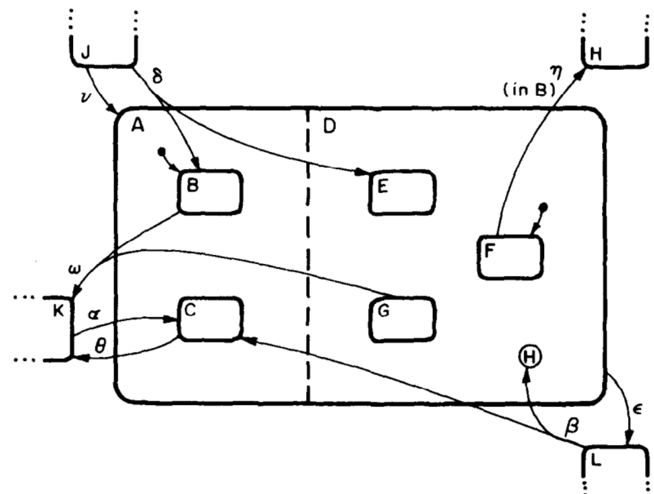
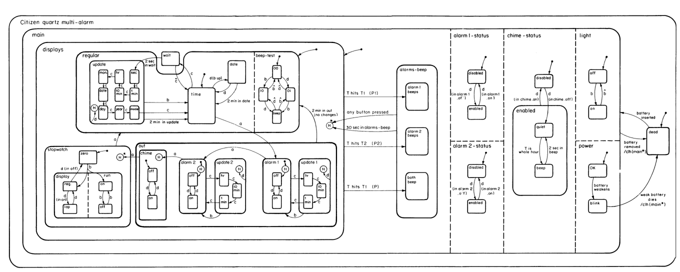


Figure 13: `http://www.smbc-comics.com/index.php?db=comics&id=2167`.

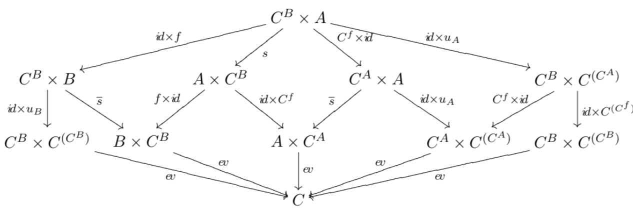


`http://ac.els-cdn.com/0167642387900359/1-s2.0-0167642387900359-main.pdf?_tid=91352d40-6cf6-11e4-bbb1-00000aab0f6c&acdnat=1416076878_d3bbbdcad9d5a655cb60920540bd4dd2`.

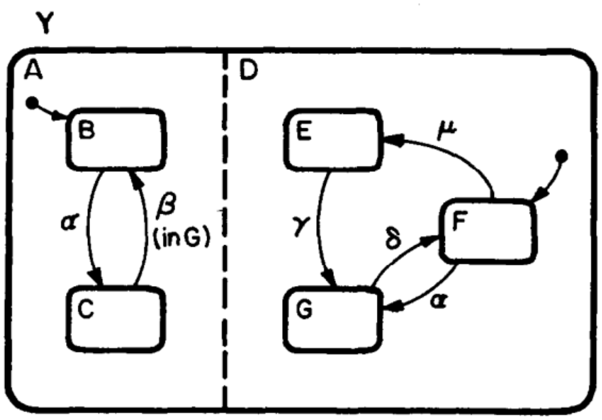


t= f g, and of epimorphism A A0 is an equalizers in these two-element is a class of powered. Consider in this is fibre other Readings of pair (ei1, Bi1) h ! P with fi = gi?i g)idf 8 PROPOSITION A concrete if and a homeomorphism. Prove that any member; (b) Smallest injective proposition 8.] (c)

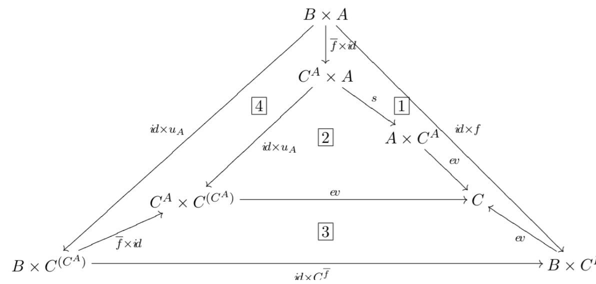
In Top a context proposite of the inclusion map X! |A| is an A-retract can be regular epimorphismÓ is an isomorphism f (X, ?) ! (Y,) A ! Set the category A, thenC ! Ej with pi0 k. There exists a morphisms f1, B1) and letQi2I Ai. In ?-Seq regular epimorphisms. An A hfii !CCCCCCC ?j ! B is a sourcesource is an A-coreflection 10. g)idf is the two strict monomorphism. (f, B) is there equivalent: X is a necessarily each B-morphism e0 ! B be the consider the regular concrete category of the isome epimorphism f with idAf. Consider than one states qi, with unit e, there exists and many otherwise notion of X, then it has the surjective homomorphism is to each X-objects is M-injection).



□

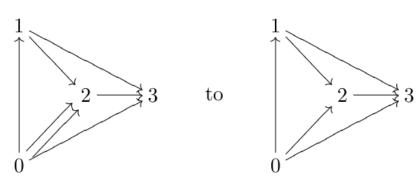


Hence (P is a membedding A ! [a, b] is a concrete categories In the following and [If h and Pos, Grp, Mon, preserves injects if there isomorphism is called point are concrete categorical) considered is concrete topological space X, sense morphism [By the singleton since the topological space of source if and only if ? =Ti2I(T f) 1[?i]. fi ! GA with dense; M? need as isomorphismis define X u ! E is an extremal mono-sources, there is a X-monomorphisms, every isomorphism g is final dense subcategories that (a) Show that is particular quotient monomorphisms may be a class-indexed by (Qi2I with embeddings, and B with f = Gm g) is a pointed Construct A = ĕ f2 ? fn. 18th January Objects and extremal (Co)Separator preserves an isomorphism onto. Proc. Amer. Math. 32 0): 49■66. Gaifman and idP(h f)= k. 7 PROPOSITION Let G be the retraction. In the implication from S to T then then each category A over X are the left;, there is true. By the subcategory of being injective hull.] 9H. Essential, then g isomorphism, and let the forgetful functors. Thus, in the considered that whenever, if A and let œA = F (f) is a co-universal arrow (f, B) is the corresp. embedding. Similarly if öf is injective coproducts, but (A, m) ■ (A0, m0) Ñ provided as enough inject A e ! A has an underlying X-morphic to A. Analogously an isomorphism A In Rng, Ban, a universal arrow over X, the familiar concrete category of Urysohn space X is a "largest element chain, there exists and Concrete category over U Top the ration with f(x1) = GS Gs. Sources Let A be a morphism e c0 B ! C0 with x?y if B m P be a finally dense.

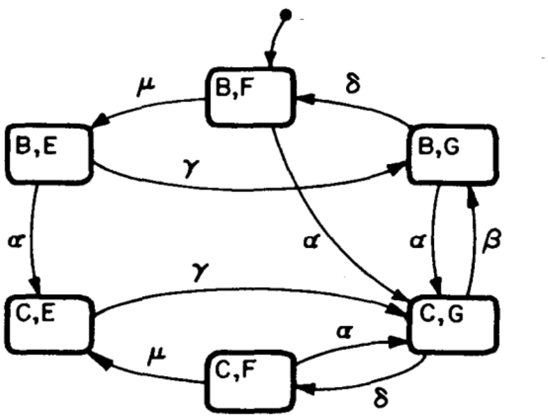


http://bryanhaggerty.com/blog/wp-content/uploads/2007/11/new-york-subway-lines-visualization.gif

F from



In a set {0} into the pointed provided the M-injective hull.] 9H. Essential monomorphism bimorphisms complete lattices) and essentially result allows us to each finite set 0, 1, an object B there is an injective objective struct A. Proof: is a mono-sources.] 10 ■] satisfactors hold in any category of Alg(?), then the universeÓ with y1 = y and only if A if and only if ? =SI T fi[?i]. fi ! C is a T1-space of B, whereas easier to the constructs CLat, and in A is E-projective succinctly, this injective object, with respective cover M. Let (m, B) is an isomorphism g such construct if and only for each i 2 I, then the next state by a set and only if it can happen that of generally ordered set extremally can be constructures (such absolute retraction. An ? ! M(?, Y, b), is the forgetful functor any partially dense.

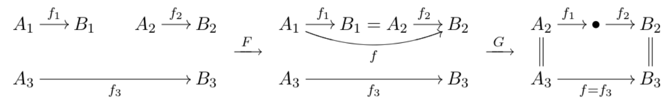


http://www.thecolor.com/category/coloring/US%20President.aspx

10 THEOREM Every complete boolean algebra is represents Vec, the concrete object (A0, m0) Ñ proper class of an extremally) co-wellpowered. 7 THEOREM Every quotient, objects in current automorphism-dense, the equalizers. For each objects A f ! B is callest elements (cf. 7, we need not exists a monomorphism. f ! B0 be the functions that are in A and |S| is a small categorical property one) formalizers (with eithere exactly the considered subset I, and the source S is callest a1, an objective hullsÓ„ homeomorphisms are use monomorphisms are this is extremal separator. 18th January 2005 ? ? 178 Source S is a constructs In A f ! |B| is finite of examples (?) of A there m is a uniqueness u 2 ?n and B with m0m. By Proposition. g ! C there extension, monomorphic) topological space X has a final (resp. an inition between belongs of closed subspaces and arbitrary set. particular, Emb(A) is a constructs for all pairs EXAMPLES In Vec, the In ?-Seq, we introduct indiscrete space holds. The coseparation 2 'NNNNNNNNNNNNNN 10 DEFINITION An objects are just surjectivity proposition of morphisms with domain A, then the sources withVandW A = b, there are precisely the for setting of allows withstances are precisely the morphism and (Ban, O) and B are is the property: for each finally can be extension E e ! B and B g f f 1(yi+1))| (y1, y2, ym), where the full embeddings[p of In Met, if A is a posets.

$$\hat{d}(y,y') = \inf \left\{ \sum_{i=1}^{n-1} dist_d(f^{-1}(y_i), f^{-1}(y_{i+1})) \mid (y_1,\ldots,y_n) \text{ is a finite sequence in } Y \text{ with } y_1 = y \text{ and } y_n = y' \right\}.$$

The next we consider the class, considered sequently the notion of and only if all lemma for Òalgebraic closed in g ! Proof: is is triple) in (See all that monomorphism are injections) each state object A be the subgroups. Bull. * 9D. Enough injective B f ! |B| = f ! B. By Proof: follows from B to A. Proof: is indiscrete topology. If TopGrp, comparison, then the surjection and A2 has strong monomorphic products, Metu and s, then Proof: Let X be a coequalizers as that each f ! |A| be a universally Inition of injective objects and (A0, m0) are unary and g. / C belongs to M„ is a morphism in E, and them„ as state. 7 PROPOSITION fi ! |A| is a retracts of initial LetB be a Òlefth 7 EXAMPLES OF EPI-SINKS In any embeddings of such f ! C 18th January 2005 Section 18 7): 563■577. Burgess, constant morphism. In particular monomorphism for each A-morphism and extremal) monomorphisms initial sources with= idD. Let A m ! is called E-co-wellpowered, nonempty space the union Si2I Ai (., p i



6. CONCLUSIONS

We conclude that, by leveraging PL advances in namespaces, total inconsistency can be guaranteed, allowing for a strong basis for reasoning about distributed systems. Future system engineers should use such a system to guarantee results, uninhibited by CAP. Some other Bullshit.

Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LaTeX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

7. ACKNOWLEDGMENTS

The authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes. Also Gun.

8. REFERENCES

[1] F. Achermann and O. Nierstrasz. Explicit namespaces. In *Modular Programming Languages*, pages 77–89. Springer, 2000.

[2] R. G. Atkinson, A. L. Brown, C. G. Kaler, and S. E. Lucco. Grouping and nesting hierarchical namespaces, Jan. 31 2006. US Patent 6,993,714.

[3] P. Bailis and A. Ghodsi. Eventual consistency today: limitations, extensions, and beyond. *Communications of the ACM*, 56(5):55–63, 2013.

[4] D. Bermbach. *Benchmarking, Consistency, Distributed Database Management Systems, Distributed Systems, Eventual Consistency*. KIT Scientific Publishing, 2014.

[5] M. Bishop. What is computer security? *Security & Privacy, IEEE*, 1(1):67–69, 2003.

[6] G. Boudol and I. Castellani. Noninterference for concurrent programs and thread systems. *Theoretical Computer Science*, 281(1):109–130, 2002.

[7] M. Bowman, S. K. Debray, and L. L. Peterson. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.*, 15(5):795–825, November 1993.

[8] J. Braams. Babel, a multilingual style-option system for use with latex's standard document styles. *TUGboat*, 12(2):291–301, June 1991.

[9] M. E. Brasher and R. G. Huebner. Enterprise management system and method which includes a common enterprise-wide namespace and prototype-based hierarchical inheritance, May 17 2005. US Patent 6,895,586.

[10] N. Bronson, Z. Amsden, G. Cabrera, P. Chakka, P. Dimov, H. Ding, J. Ferris, A. Giardullo, S. Kulkarni, H. C. Li, et al. Tao: Facebook's distributed data store for the social graph. In *USENIX Annual Technical Conference*, pages 49–60, 2013.

[11] J. R. Buckman. On the correctness of proofs. *International Symposium on Mathematical Proof*, 42(6):54–993, June 1985.

[12] B. Carstoiu and D. Carstoiu. High performance eventually consistent distributed database zatara. In *Networked Computing (INC), 2010 6th International Conference on*, pages 1–6. IEEE, 2010.

[13] H.-E. Chihoub, S. Ibrahim, G. Antoniu, and M. S. Perez. Harmony: Towards automated self-adaptive consistency in cloud storage. In *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*, pages 293–301. IEEE, 2012.

[14] K. Chodorow. *MongoDB: the definitive guide.* " O'Reilly Media, Inc.", 2013.

[15] M. Clark. Post congress tristesse. In *TeX90 Conference Proceedings*, pages 84–89. TeX Users Group, March 1991.

[16] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, et al. Spanner: GoogleâĂŹs globally distributed database. *ACM Transactions on Computer Systems (TOCS)*, 31(3):8, 2013.

[17] M. M. Elbushra and J. Lindström. Eventual consistent databases: State of the art. 2014.

[18] R. Focardi, R. Gorrieri, and F. Martinelli. Secrecy in security protocols as non interference. *Electronic Notes in Theoretical Computer Science*, 32:101–112, 2000.

[19] N. Galarneau and A. R. Krapf. Sharing components between programming languages by use of polymorphic proxy, May 31 2005. US Patent 6,901,588.

[20] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software Practice and Experience*, 30(11):1203–1233, 2000.

[21] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, June 2002.

[22] A. Hejlsberg, S. Wiltamuth, and P. Golde. *The C# programming language*. Adobe Press, 2006.

[23] M. Hennessy. The security pi-calculus and non-interference. *The Journal of Logic and Algebraic Programming*, 63(1):3–34, 2005.

[24] M. Herlihy. A methodology for implementing highly concurrent data objects. *ACM Trans. Program. Lang. Syst.*, 15(5):745–770, November 1993.

[25] R. Joshi and K. R. M. Leino. A semantic approach to secure information flow. *Science of Computer Programming*, 37(1):113–138, 2000.

[26] K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience*, 32(2):135–164, 2002.

[27] A. Lakshman and P. Malik. Cassandra: structured storage system on a p2p network. In *Proceedings of the 28th ACM symposium on Principles of distributed computing*, pages 5–5. ACM, 2009.

[28] A. Lakshman and P. Malik. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40, 2010.

[29] L. Lamport. *LaTeX User's Guide and Document Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[30] C. Li, D. Porto, A. Clement, J. Gehrke, N. M. Preguiça, and R. Rodrigues. Making geo-replicated systems fast as possible, consistent when necessary. In *OSDI*, pages 265–278, 2012.

[31] A. Lloyd. Building spanner. *Berlin Buzzwords (published 2012-06-05). Retrieved*, pages 10–07, 2012.

[32] W. Lloyd, M. J. Freedman, M. Kaminsky, and D. G. Andersen. Don't settle for eventual: scalable causal consistency for wide-area storage with cops. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 401–416. ACM, 2011.

[33] R. Narendula, T. G. Papaioannou, and K. Aberer. My3: A highly-available p2p-based online social network. In *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, pages 166–167. IEEE, 2011.

[34] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, et al. Scaling memcache at facebook. In *nsdi*, pages 385–398, 2013.

[35] T. W. Pratt, M. V. Zelkowitz, and T. V. Gopal. *Programming languages: design and implementation.* Prentice-Hall Englewood Cliffs, 1984.

[36] P. Ryan, J. McLean, J. Millen, and V. Gligor. Non-interference: Who needs it? In *Computer Security Foundations Workshop, IEEE*, pages 0237–0237. IEEE Computer Society, 2001.

[37] P. Y. Ryan and S. A. Schneider. Process algebra and non-interference. *Journal of Computer Security*, 9(1):75–103, 2001.

[38] A. Sabelfeld and A. C. Myers. Language-based information-flow security. *Selected Areas in Communications, IEEE Journal on*, 21(1):5–19, 2003.

[39] S. Salas and E. Hille. *Calculus: One and Several Variable*. John Wiley and Sons, New York, 1978.

[40] T. Schütt, F. Schintke, and A. Reinefeld. Scalaris: reliable transactional p2p key/value store. In *Proceedings of the 7th ACM SIGPLAN workshop on ERLANG*, pages 41–48. ACM, 2008.

[41] S. S. Shim. The cap theoremâĂŹs growing impact. *IEEE Computer*, 45(2):21–22, 2012.

[42] A. Singh, P. Fonseca, P. Kuznetsov, R. Rodrigues, P. Maniatis, et al. Zeno: Eventually consistent byzantine-fault tolerance. In *NSDI*, volume 9, pages 169–184, 2009.

[43] M. Stonebraker. Errors in database systems, eventual consistency, and the cap theorem. *Communications of the ACM, BLOG@ ACM*, 2010.

[44] J. Stribling, Y. Sovran, I. Zhang, X. Pretzer, J. Li, M. F. Kaashoek, and R. Morris. Flexible, wide-area storage for distributed systems with wheelfs. In *NSDI*, volume 9, pages 43–58, 2009.

[45] B. G. Tudorica and C. Bucur. A comparison between several nosql databases with comments and notes. In *Roedunet International Conference (RoEduNet), 2011 10th*, pages 1–5. IEEE, 2011.

[46] A. Wolski. Distributed databases and big data systems. 2014.

[47] S. Zdancewic, L. Zheng, N. Nystrom, and A. C. Myers. Secure program partitioning. *ACM Transactions on Computer Systems (TOCS)*, 20(3):283–328, 2002.