# 3 things I wish I knew when I started designing languages

#### About "poor me"



#### About the real me

Flunked trig, flunked chem, never took calculus or physics.

Graduated HS with a 2.8 GPA

Bachelor of Arts in English Literature

3 years as an editor; 2 as a DBA; 5 as a software engineer before grad school

I am not nor was I ever a PL researcher

## This talk is about me (designing a language)

- 1. Misgivings: how I almost never began
- 2. Lucky guesses: things I got right
- 3. Discoveries: stuff I learned along the way

## Prelude: misapprehensions and misgivings

how we FUD ourselves out of language design

## 1: The Look

## An audacious new language should look unique!

#### 2PC

{request, Xact, Client} -> if (valid(request)) then multicast(Members, {prepare, Xact, Myaddress}); end if {prepare, Xact, Coordinator} -> if (exec(Xact)) then send(Coordinator, {vote, Xact, y, Myaddress}); else send(Coordinator, {vote, Xact, n, Myaddress}); end if {vote, Xact, Vote, Agent} -> // voting... if (Vote == n) then send(Members, {status, Xact, abort}); else // local state, etc. if () then end end if {status, Xact, Status, Coordinator} -> if (Status == abort) then abort(Xact); else commit(Xact); end if send(Coordinator, {ack, Xact, Myaddress});

{ack, Xact, Agent} ->

## 1: The Need

some bicyclist[sic] @palvaro Replying to @palvaro @cmeik	the heart rears wings bold and 🧇	~
2: you're neter alvaro?	last year carl hewitt honored me by wall	king
<ul> <li>A jerf on Apr 8, 2011 [-]</li> <li>I get where you are coming from, and it's an optimistic interpretation of JRuby's perf</li> <li>I don't get the idea that some people seen see a great deal of sloppiness around performed and the statement of the second statement of the se</li></ul>	a good plan, as long as the plan is to eventually fully detach from Ruby. Being even ormance, is still starting from a terrible position in so many ways. In to have that performance doesn't matter for distributed systems, when the truth is prmance, because it doesn't really matter that much. Small servers or small clusters	two or three times as fast as Ruby, which seems to be s the exact opposite. Desktops and even cell phones, we s, we still say throw more hardware at it and just hack
maybe bytes of RAM, but there is somethin 4 PM 2600 programmer. (Facebook apparently p later is a good idea, they may well iterate	en you're serious about distributed systems is also when you are counting every one ig you are obsessing over. And maybe you're obsessing over more than one of thess bublished the specs for their machines today. Tell me they aren't too concerned about their way into a cool abstraction that will <i>never</i> perform. Designing a distributed sys ing a new 3D framework without worrying about performance not necessarily a fata 11:45 PM - 27 Apr 2013	se at once, all with an intensity that would credit an Atari ut performance.) I'm not sure leaving performance for stem abstraction without worrying about performance
1 121 ()7 ili	the heart rears wings bold and <	h a library?
Some bicyclist[sic] @@palvaro · 16 Sep 2016	(roughly transcribed) OH: "in racket, everything is parenthesis. what is the thing in	How is this different from elixir?
P: *pointing derisively* HAHAHAHAHAHAHAHAHAHA > 5 t]	your language that is everything and that I don't buy?	
some bicyclist[sic] @ @palvaro · 16 Sep 2016 (HPTS 2009)		
♀ ▲ CoffeeDregs on Apr 8, 201:	L [-]	
I like it! umm but what	t the hell is it? I STFA (skimmed-the-f*ing-article) and don't know	what's going on here. Quick! What does this mean

## 1: The Impact



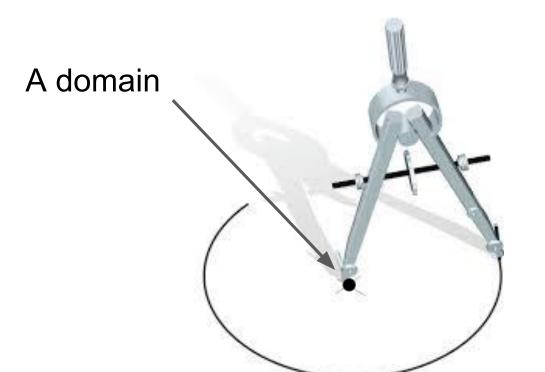
## Lucky guesses: things I got right

## Lucky guess 1:

context context

## **Every language is a DSL**

#### Thy firmness makes my circle just



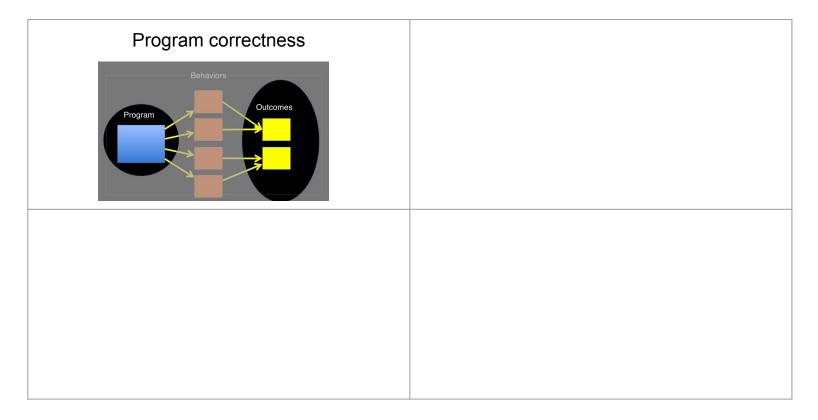
## More lucky guesses: a damn problematic domain

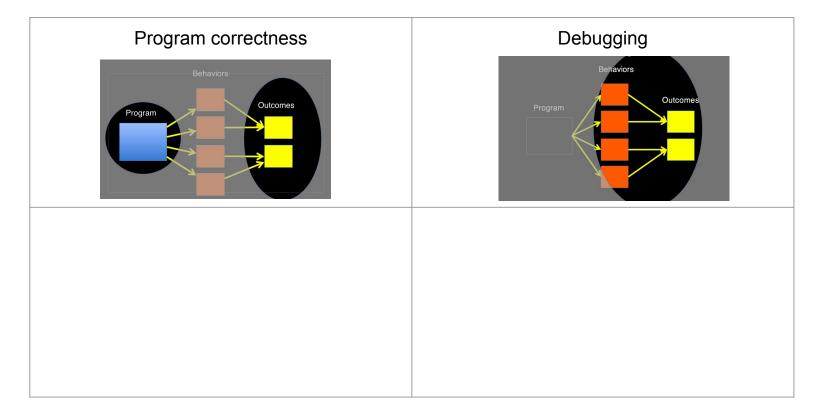
$$P_{real}(success) = F\left(\frac{perceived\ crisis}{perceived\ pain\ of\ adoption}\right)$$

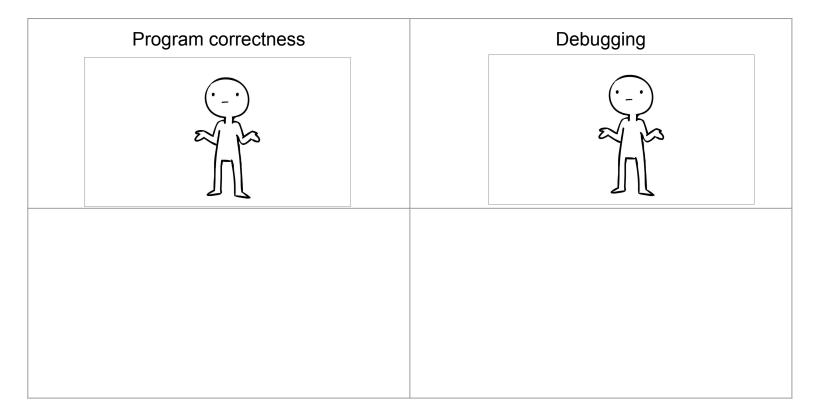
## Hiding and illuminating

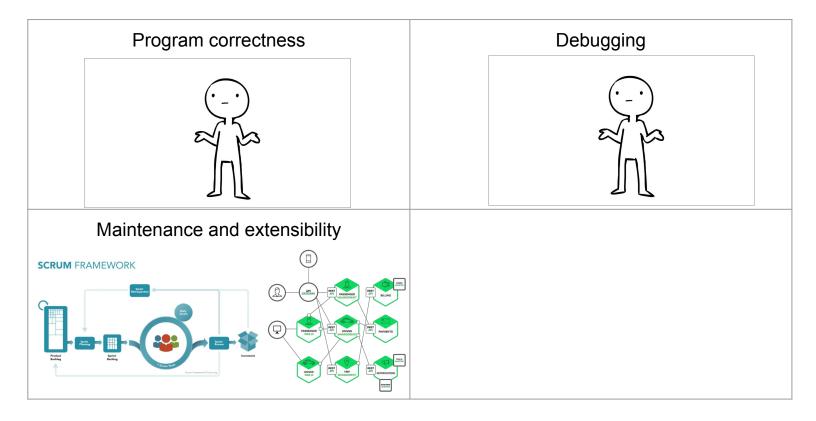
N

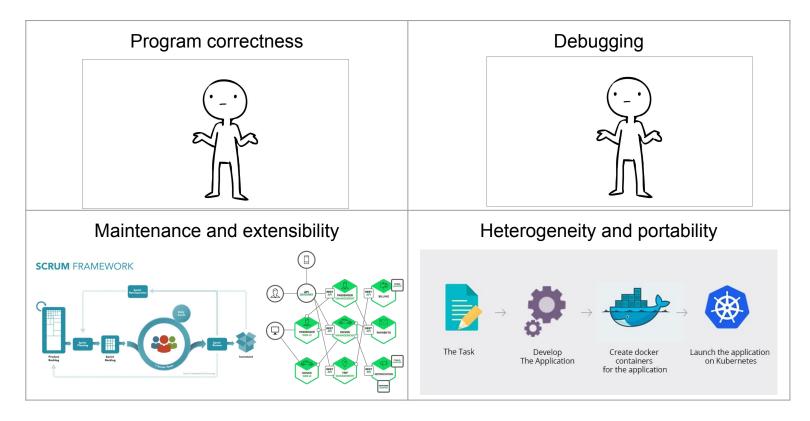
N







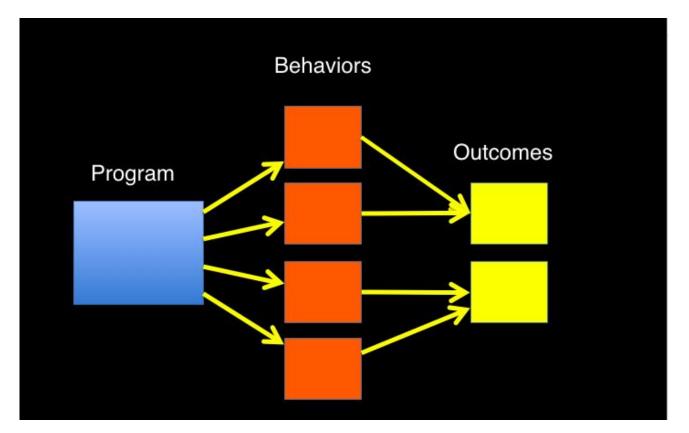




## Rearranging the deckchairs...



## Why so damn hard?



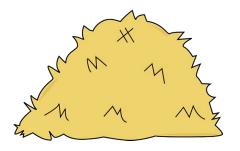
## The right language would focus our attention on

How **data** flows through the system;

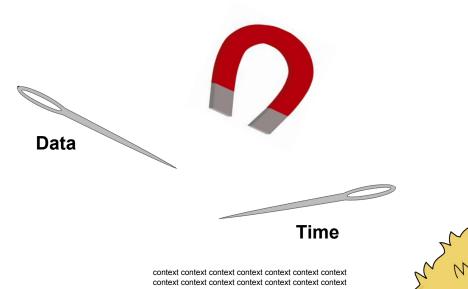
How it is allowed to change over **time**;

Where and when we can control how it changes and when we can't.

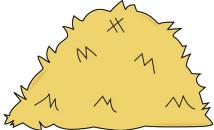
Everything else, arguably, is a distraction



Control flow

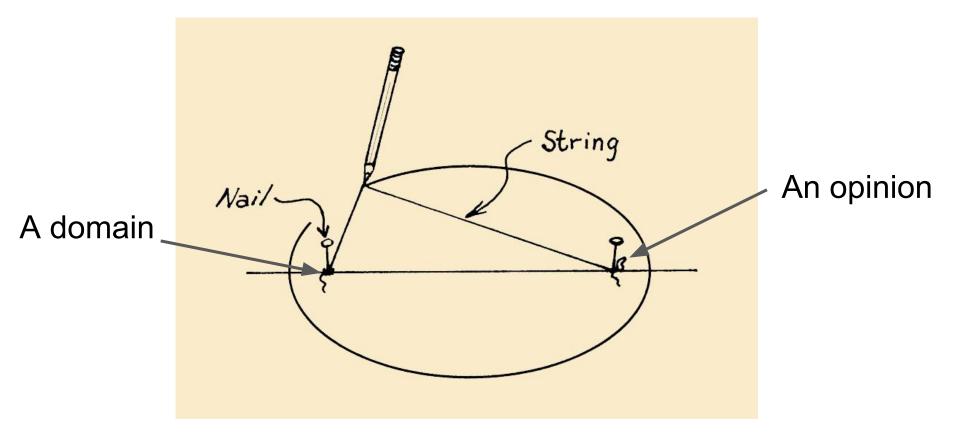


context contex



State representation

#### Thy firmness makes my circle just



#### Many moons passed

so I'm sitting in the bath reading lamport, like I do most nights. and it occurs to me...

are we thinking about this wrong, trying to figure out how to map the doughface language onto an overwriting implementation? it occurs to me that we should have this:

- all facts (instantaneous events) are stored (almost) eternally by appending to a log. this log is timestamp order, obviously, but we could imagine indexes into it.
- all a program does is define what is true when, based on what is already true then. a
  predicate p is true at a given time N if it was a fact given at that time (cheap to look
  up since our log is sorted), or if it can be proven that a tuple has carried over from a
  previous time (i.e., if (\exists a tuple A \in p@M s.t. M < N) and (~\exists a tuple B \in
  del\_p@O s.t. M < O < N).</li>

statement structure:

Vol1, p. 23

"encapsulation [...] appears antithetical to declarativeness.

backing up. what are reasonable state primitives for a programming language?

talk about the data: name it, talk about how it changes.

This is all very well and good in a datalog program, which is evaluated over a static set of ground facts until there are no more conclusions to be drawn. But when we introduce the notions of time and communication, from pure logic to asynchronous distributed systems, we feel uncomfortable (understandably) with the idea of rules firing in no particular order.

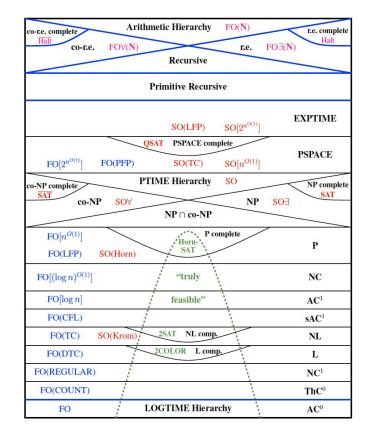
## Inspiration

In the beginning, there were two measures of computational complexity: time and space. From an engineering standpoint, these were very natural measures, quantifying the amount of physical resources needed to perform a computation. From a mathematical viewpoint, time and space were somewhat less satisfying, since neither appeared to be tied to the inherent mathematical complexity of the computational problem.

In 1974, Ron Fagin changed this. He showed that the complexity class NP — those problems computable in nondeterministic polynomial time — is exactly the set of problems describable in second-order existential logic. This was a remarkable insight, for it demonstrated that the computational complexity of a problem can be understood as the richness of a language needed to specify the problem. Time and space are not model-dependent engineering concepts, they are more fundamental.

$$\Phi_{3\text{-color}} \equiv (\exists R^1)(\exists Y^1)(\exists B^1)(\forall x) \Big[ \big( R(x) \lor Y(x) \lor B(x) \big) \land (\forall y) \Big( E(x,y) \rightarrow (R(x) \land R(y)) \land \neg (Y(x) \land Y(y)) \land \neg (B(x) \land B(y)) \Big) \Big]$$

 $\Phi_{\text{SAT}} \equiv (\exists S)(\forall x)(\exists y)((P(x,y) \land S(y)) \lor (N(x,y) \land \neg S(y))) .$ 

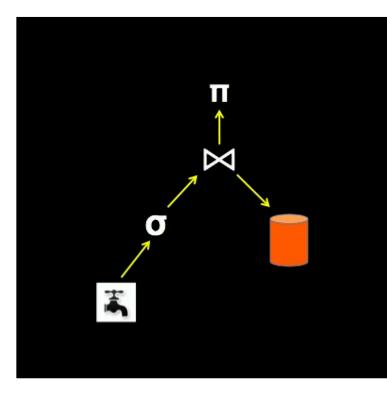


Although few programmers consider their work in this way, a computer program is a completely precise description of a mapping from inputs to outputs. In this book we follow database terminology and call such a map a *query* from input structures to output structures. Typically a program describes a precise sequence of steps that compute a given query. However, we may choose to describe the query in some other precise way. For example, we may describe queries in variants of first- and second-order mathematical logic.

Fagin's Theorem gave the first such connection. Using first-order languages, this approach, commonly called descriptive complexity, demonstrated that virtually all measures of complexity can be mirrored in logic. Furthermore, as we will see, the most important classes have especially elegant and clean descriptive characterizations.

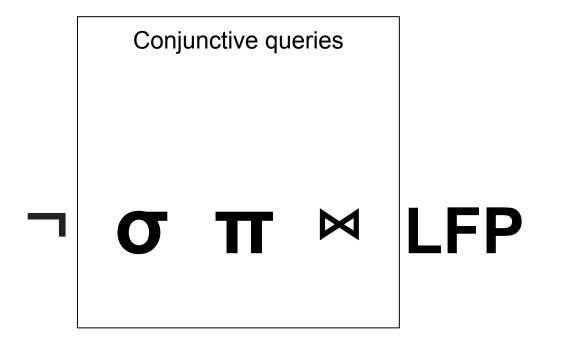
#### Queries made a neat lens...

create view response as select client, server, code, document from request r, page p where r.server = p.server and r.URI = p.URI;

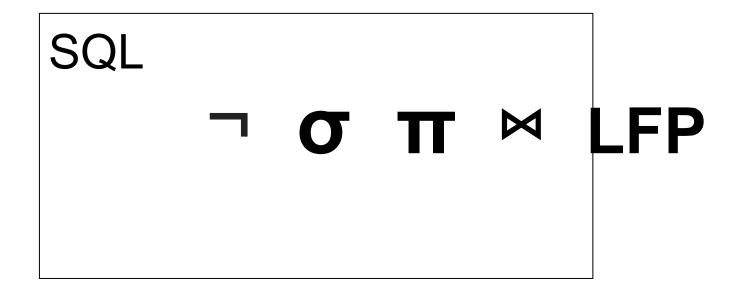




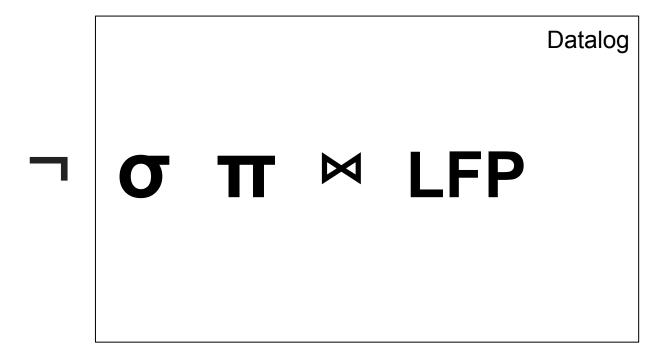
## fragments



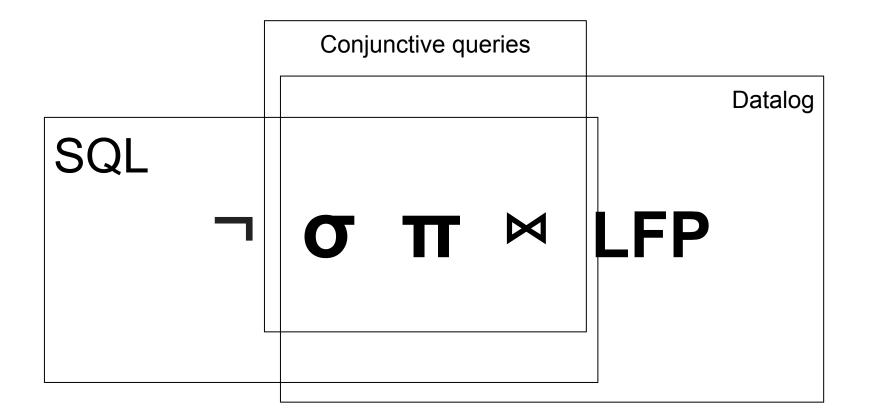
## fragments



## fragments



## Or maybe they are lassos



context context

# knowledge("details")

knowledge(host1, "details")

Contexualized by location (space)

#### Datalog cannot express

Mutable state

Uncertainty

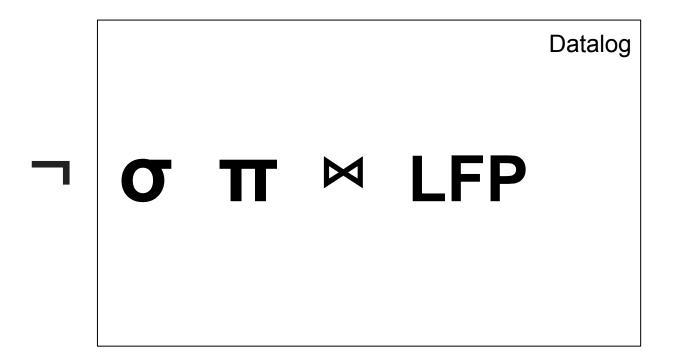
# knowledge(host1, "details", 27)

Contexualized by relative order (time)

## register(host1, "current value", 27)

## kvs(host1, key, "current value", 27)

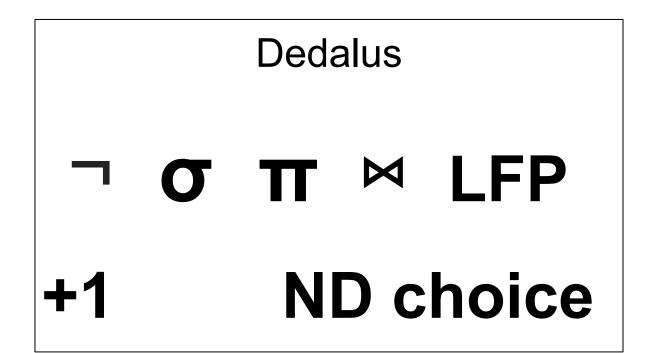
#### Or maybe they are lassos



#### Or maybe they are lassos



#### Dedalus can express it all. but...



#### Paxos

		- 14 44
Dedalus al	<pre>nodes(A, N, I)@next :- nodes(A, N, I); seed(A, S)@next :- seed(A, S), notin update_seed(A);</pre>	bn't want to
	<pre>seed(A, S+C)@next := seed(A, S), update_seed(A), agent_cnt(A, C);</pre>	
_		
covr roqui	<pre>prepare(B, A, S, M)@async :- proposal(A, M), seed(A, S), nodes(A, B, _); update_seed(A) :- proposal(A, _);</pre>	her not use
say; requi		
<b>J</b> , I	redo(A, M) :- timeout(A, M), notin accepted(A, _, M);	
2PC	<pre>prepare(B, A, S, M)@async :- redo(A, M), seed(A, S), nodes(A, B, _); timer_svc(A,M,3) :- redo(A, M);</pre>	3PC
260	update_seed(A) :- redo(A, M);	
<pre>cancommit(Agent, Coord, Xact)@async :- begin(Coord, Xact), agent(Coord, Age</pre>		
<pre>vote_msg(Coord, Agent, Xact, "Y")@async :- cancommit(Agent, Coord, Xact), c</pre>		
<pre>vote(C,A,X,S) :- vote_msg(C,A,X,S);</pre>	<pre>response_log(C, A, S, O, M)@next :- response_log(C, A, S, O, M);</pre>	<pre>d(A); , agent_cnt(A, C);</pre>
<pre>timer_svc(A, X, 4) :- cancommit(A, _, X);</pre>	// workaround for the fact that c4 can't count strings!	ed(A, S), nodes(A, B, _);
<pre>// the coordinator is the distinguished node that is not an agent</pre>	<pre>//response_cnt(C, S, count<a>) :- response_log(C, A, S, 0, 0s);</a></pre>	u(n, 5), housen, 5, _/,
abort(A, X)@next :- timeout(A, X), notin coordinator(A, A), notin precommit	<pre>response_cnt(C, S, count<i>) :- response_log(C, A, S, O, Os), nodes(C, A, I);</i></pre>	_, M);
	<pre>best(C, S, max&lt;0s&gt;) :- response log(C, A, S, O, Os);</pre>	, S), nodes(A, B, _);
authorized(C, X) :- vote_msg(C, _, X, "Y"), notin missing_vote(C, X), notir		
	<pre>//agent_cnt(C, count<i>) :- nodes(C, _, I);</i></pre>	e(C, A, S, O, Os);
<pre>precommit(A, C, X)@async := authorized(C, X), agent(C, A); ack(C, A, X)@async := precommit(A, C, X), prepared(A, C, X, "Y");</pre>	<pre>agent_cnt(C, count<i>) :- what(C, I);</i></pre>	g(C, A, S, O, M);
<pre>timer_cancel(A, X) :- precommit(A, _, X), prepared(A, _, X, "Y");</pre>	accept(A, S, 0)@asvnc :- agent cnt(C, Cnt1), response cnt(C, S, Cnt2),	strings! C, A, S, O, Os);
<pre>timer_svc(A, X, 4) :- precommit(A, C, X), prepared(A, C, X, "Y");</pre>	response_log(C, _, S, 0, 0s), best(C, S, 0s), nodes(C, A, _), 0s != 1, Cnt2 > Cnt1 / 2;	A, S, O, Os), nodes(C, A, I);
<pre>//commit(A, X)@next :- timeout(A, X), precommitted(A, C, X), notin abort(A,</pre>	<pre>accept(A, S, P)@async :- agent_cnt(C, Cnt1), response_cnt(C, S, Cnt2), response_log(C, _, S, 0, 0s),</pre>	Os);
<pre>commit(A, X) :- timeout(A, X), precommitted(A, C, X), notin abort(A, X);</pre>	<pre>best(C, S, Os), my_proposal(C, P), nodes(C, A, _), Os == 1, Cnt2 &gt; Cnt1 / 2;</pre>	
<pre>precommitted(A, C, X) :- precommit(A, C, X);</pre>		
<pre>precommitted(A, C, X)@next :- precommitted(A, C, X);</pre>	// acceptor	<pre>sponse_cnt(C, S, Cnt2), response_log(C, _, S, 0, 0s), best(C, S, 0s), nodes(C, sponse_cnt(C, S, Cnt2), response_log(C, _, S, 0, 0s), best(C, S, 0s), my_propo</pre>
<pre>abort(C, X)@next :- vote(C, _, X, "N");</pre>	<pre>dominated(A, S) := prepare(A, _, S, _), prepare_log(A, S2, _), S2 &gt; S;</pre>	sponse_cnr(c, s, cnr2), response_rog(c, _, s, o, os), besr(c, s, os), my_propo
<pre>commit(C, X)@next :- ack(C, _, X), notin missing_ack(C, X), notin abort(C,</pre>	<pre>can_respond(A, C, S, M) :- prepare(A, C, S, M), notin dominated(A, S); prepare_response(C, A, S, 0, 0s)@async :- can_respond(A, C, S, M), accepted(A, 0s, 0), highest_accepted(A, 0s);</pre>	
<pre>missing_ack(C, X) :- agent(C, A), running(C, X), notin ack(C, A, X);</pre>	prepare_response(C, A, S, "anything", 1)@async :- can_respond(A, C, S, M), notin accepted(A, _, _);	_log(A, S2, _), S2 > S; notin dominated(A, S);
<pre>missing_vote(C, X) := agent(C, A), running(C, X), notin vote(C, A, X, "Y");</pre>		espond(A, C, S, M), accepted(A, Os, O), highest_accepted(A, Os);
<pre>prepared(A, C, X, "Y") :- cancommit(A,C,X), can(A,X);</pre>	highest_accepted(A, max <s>) :- accepted(A, S, _);</s>	<pre>:- can_respond(A, C, S, M), notin accepted(A, _, _);</pre>
prepared(A, C, X, Y)@next :- prepared(A,C,X,Y);	accepted(A, S, M) :- accept(A, S, M);	0;
	accepted(A, S, M)@next :- accepted(A, S, M);	
<pre>timer_svc(C, X, 5) :- begin(C, X);</pre>		
<pre>abort(C, X)@next := timeout(C, X), coordinator(C, C), missing_ack(C, X), no</pre>	prepare_log(A, S, M) :- prepare(A, _, S, M);	
	prepare_log(A, S, M)@next :- prepare_log(A, S, M);	M);
<pre>commit(A, X)@async :- commit(C, X), agent(C, A), notin abort(C, A);</pre>	<pre>my_proposal(A, P) :- proposal(A, P);</pre>	<b>→</b>
<pre>abort(A, X)@async :- abort(C, X), agent(C, A);</pre>	""	

Waiting requires counting

Counting requires waiting

(Joe Hellerstein)

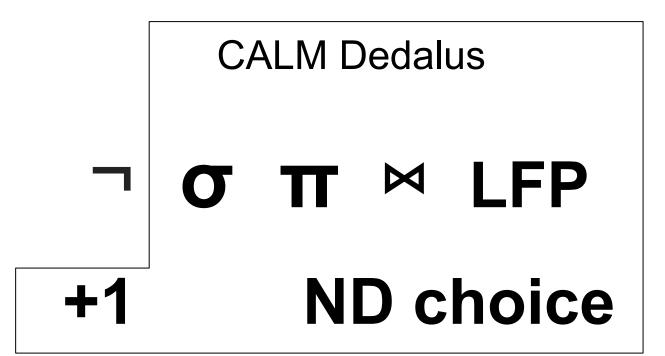
Waiting requires counting

#### Nonmontonicity required to express coordination

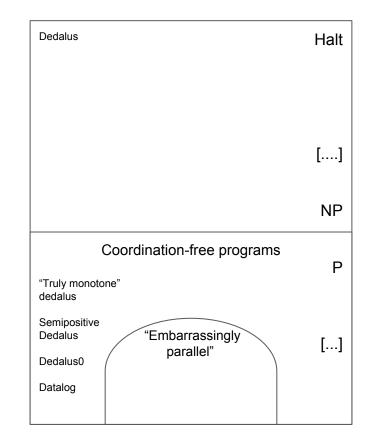
Counting requires waiting

**Coordination required to tolerate nonmonotonicity** 

#### Or maybe they are lassos

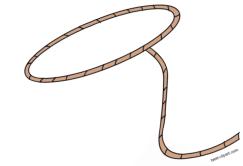


## Pop descriptive complexity

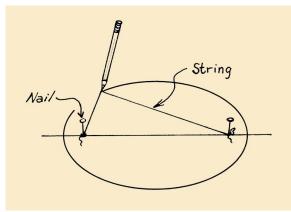


#### Languages and the design process

context contex







#### Languages and the design process

Subject



Object

#### Languages and the design process







The look

The look

it's about the fit

**The look** it's about the *fit* 

The need

**The look** it's about the *fit* 

The need

it's about *our* need

**The look** it's about the *fit* 

The needit's about our need

The impact we

well....

#### **DEDALUS: Datalog in Time and Space**

#### Peter Alvaro- William R. Marczak- Neil Conway-Joseph M. Hellerstein- David Maier Russell Sears-<sup>1</sup>University of California, Berkeley <sup>1</sup>Portland State University

{palvaro,wrm,mrc,hellerstein,sears}4cs.berkeley.eds maier6cs.pdm.edu

#### ABSTRACT

ABSTRACT Receipt course has explored using During based languages to ex-properties of determinant systems proved afficial to account in During, Free, the actic of any activity provided official to account in During, Free, the actic of activity and prove more shown in accounts, become, the output official to account in the account based of the activity of the activity of activity is tacked applies, buy cancersize, solutions from the activity of the activity of the language of an improvement, activity of activity of the language of an improvement, and these applies to complete the activity of the language of a improvement, activity of the activity of the language of an improvement, and these properties from the more shown in the activity of the language of

enter tot for a draging, "It is ingrated as a final set of pripar-bility of the strength of the strength of the strength of the therm private integrates are strength of the the strength of the strengt of the strength of the strength of the strength of t

#### 1. INTRODUCTION

<text><text><text><text> In recent years, there has been a resurgence of interest in Datain recent years, that that being the analysis of the second secon

stratification. We begin by defining Densaras, a restricted sublanguage of Dat-alogs (Section 2). We show how Denautos, supports state update in Section 3, prove temporal safety and stratifiability properties of Dansuras, in Section 4, and describe a simple, efficient evalua-<sup>1</sup>Ditacci is introduce an processor language for Himms, high bool lan-guage for programming distributed systems that will replace Overlag in the BOOD processor [20], so such, all index from the sharener brighten the BOOD processor [20]. So such, all index from the sharener brighten these of the sover's hown. Lenged Blooss. The distance Integration theory of the sover's hown. Lenged Blooss. The distance Integration and Enter of Lenge Vision and States and States and States and Enter of Lenge Addition and the solution of the Costs dogstone and Enter of Lenge Addition and the solution of the solution in the solution means in firstly proceeding.

sulting languages have been promoted for their compact and natural

Lineage-driven Fault Injection				
Peter Alvaro UC Berkeley pelvaro@cs.berkeley.edu	Joshua Rosen UC Barkeley roserwille@gmail.com	Joseph M. Hellerstein UC Berkeley hellerstein@cs.berkeley.edu		
ANSTRACT Table is to begin support. It is beginned of an any method of the second second second second second and the second second second second second second the second seco	suggement sys- colors and con- dense. Were sub- dense. Were sub- dense. Were sub- dense. Were sub- section and con- tained sub- tional sub- section and sub- se	metricing are explores architectures with well and explores of the 4.4 sectors of the sector of the		
Categories and Subject Descriptors 31.2.4 [Database Management]: Systems—Disrib	uted Databases combinations	hence are unlikely to find rare once conditions caused by complex combinations of failures. Wone still, fault injection techniques— regardless of their search strategy—causes effectively guarantee		
Keywords fash tolonace; verification; prevenance	work as FRE breatables to a statagies can	coverage of the space of pensible failure scenarios. Primeworks such as FME [36] are a combination of Pente-force search and heuristics to guide the concernition of faults: such heuristic search stanging can be officient at successing are failure scenarios, bar,		
1. INTRODUCTION The bit denses to a critical future of modern do yotens, which are often duritheted is accorated atom (2, 12, 10, 14, 26, 55, 88). Furth softenet prov- tical, instanting and data seguration (4, 56, 78), and which is subsidiary data and the second second second in the context of these modern architectures. With so many machinestance three which is to choose to take a bottom-up approach to data management.	In menogeneral to resource data resource data resoure data resource data resource data res	manch, they do limb to item the upper of possible proton unbiated for arrange that for the borded and systems opposen sciences for state fash world sends the systems opposens convertige state fash world sends the networks of the state of the state of the state of the state of provide senses. The statisty about the state of the state of the state of the state of the state of the state of states of the state of the state of the state of states of the state of states of the states and the st		
Prevaiusion to make digital or hand appire of past or all of this to classroom one's generated withins the previated their replex are not to profit or conversional advantage, and their appire to an outside or a the first page. Copyrights for third paper programmers based. The first maps of the mass, customs the sensetimation. Copy BioMoNP 11: May 13-Jane A 2015, Methousen, Neural, Annual ADA W 15: 1-4602 (2014) 01200.	white present or make or disablend for and the full or first work must be optimized by the full of the state of the full of the full of the full of the state of the full of the full of the full of the full of the full of the full	interactions, we propose a novel log-down storingy for time goals, we propose a novel log-down storingy for spectra (LJPE). (EDF) is inspiced by the database 18- of data biology (17,25,56,56,25,98), which indexs it concert system storics to the data and monogoe that LJPT one dura lineage to reason backsouch fyrms ef- al short whether a given concert contenues need have		

<section-header><section-header><section-header><text><text><text><text><text><text><text><text><text><text></text></text></text></text></text></text></text></text></text></text></section-header></section-header></section-header>						
<text><text><text><section-header><text><text><text><text><text><text><text></text></text></text></text></text></text></text></section-header></text></text></text>	Growing a protocol					
<text><text><text><text><text><text><text><text><text><text><text></text></text></text></text></text></text></text></text></text></text></text>	Kamala Ramosubeamanian <sup>1</sup> , Katheyu Dahlgren <sup>1</sup> , Asha Karim <sup>1</sup> , Sanjana Maiya <sup>1</sup> , Sarah Borland <sup>1</sup> , Booz Lookos <sup>1</sup> , Peter Alvarn <sup>1</sup>					
<text><section-header><text><text><text><text><text><text><text></text></text></text></text></text></text></text></section-header></text>						
<text><text><text><text><text><text><text><text></text></text></text></text></text></text></text></text>	{kramasub, kmdahlgr, akarim, smaiya, sborland, palvaro}@ucsc.edu					
	<text><section-header><section-header><text></text></section-header></section-header></text>	with an experimentation. With a strategy of the stra				

#### Fixed It For You:

<sup>1</sup>Universite of California, Santa Ceux. <sup>2</sup>Technische Universität Berlin

	Abstract	venues such as CIDB!) that provenance could	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Distriction optimum are difficult to regarm and in an imaginary method to obleg distribution that has non-indigate and the obleg distribution of the strength optimum and the interaction of a strength optimum and the strength optimum and the strength optimum and the strength optimum distribution of the strength optimum and the strength distribution of the strength optimum and the strength distribution of the strength optimum and the strength distribution of the strength optimum and the strength op- diment of the strength optimum and the strength optimum distribution of the strength optimum and the strength optimum distribution of the strength optimum and the strength optimum distribution optimum. The strength optimum and the strength the strength optimum and the strength optimum and the strength the strength optimum and the strength optimum and the strength the strength optimum and the strength optimum and the strength the strength optimum and the strength optimum and the strength the strength optimum and the strength optimum and the strength the strength optimum and the strength optimum and the strength optimum and the strength optimum and the strength	Insuit of the debugging couldness there if the fitters: []. Easing generations have if debugging a price place diversion as an entrin wave versage for a field place diversion as an entrin wave versage for a field place diversion as a superfitting and the sec- mentations. Here, we are an appendix the sec- state, a field y coefficient in the sec- mentation of the sec- tion of the sec- mentation of the sec- tion of the sec- tion of the sec- mentation of the sec- tion of the sec- matical sec- matical sec- tion of the sec- tion of the sec- matical section of the sec- tion of the sec- matical section of the sec- tion of the sec- station of the sec- station of the sec- tion of the sec- tion of the sec- tion of the section of the sec- tion of the section of the sec	
	1. INTRODUCTION	ability and durability in the face of faults. Sta processance-based debugging is of no help to the	
	Described systems presents or well with the two models of the two stars of	Less, because them is an idealizing lass of containers of the second se	
	This acticle to published under a Dentityer Demonstra Attention License InterphytrathysemonococqUerophysiology (5.1), which permits derivations and reproduction is any machine are well as delaying derivative product pro- sede that your relative the engineer levels to the anterbasets and CTBR (500). 9th Menual Conference on Research Policy Systems Research (CMR: 70) Menuary 15 h (2007), Ackinger, Collifornia, USA:	correct the datributed program towards coupli sequential specification. These program modifi- amount to a few flaws of code, avoiding the co- combinatorial program synthesis.	

#### Consistency Analysis in Bloom: a CALM and Collected Approach

#### Peter Alvaro, Neil Conway, Joseph M. Hellerstein, William R. Marczak {palvaro, nrc, hellerstein, wrm}@cs.berkeley.edu University of California, Berkeley

#### ABSTRACT

which is part is the first transmission of the state of Distributed programming has become a topic of widespread interest, and many programmers now wrontle with tradeoffs between data and many programmers now wrands with tradeoffs between data consistency, availability and latency. Distributed transactions are often rejected as an undersinable tradeoff oday, but in the absence of transactions there are few concerns principles or tools to help programmers design and verify the correctness of their applications. We address this situation with the CALM principle, which cor-We address the stratation with the CAM principle, which con-cers the fields of dimensional programs tools for legical measurements of the stratage of the stratage of the stratage language that is assessible to highly hered consistency and principle measurements on the strategistic programmatical program and independent of the stratage of the stratage in RMy, which independent on the stratage of the stratage in RMy and of solid programs and the stratage in RMy and the strategistic programs and the stratage in RMy and the independent of the stratage of the stratage in RMy and the independent of the stratage of the stratage in RMy and the independent of the stratage in RMM and the stratage in RMM and the independent of the stratage independent of the stratage in RMM and the stratage in the stratage in RMM and the stratage in RMM and the independent of the stratage independent of the stratage in RMM and the independent of the stratage independent of the stratage in RMM and the independent of the stratage independent of the stratage in RMM and the independent of the stratage independent of the stratage in RMM and the independent of the stratage independent of the stratage in RMM and the independent of the stratage independent of the stratage in RMM and the independent of the stratage independent of the stratage in RMM and the independent of the stratage independent of the stratage in RMM and the independent of the stratage independent of the stratage in RMM and the stratage independent of the stratage in RMM and the stratage independent of the stratage in RMM and the

1. INTRODUCTION

#### rebut flowy and protein level to hidy regumment means about and manage high level program properties in the face of loosily requires in this disregard. The properties is the face of logical programs in this disregard. Our approach is based on the as only programs in this disregard. Our approach is based on the as only both starts apply and arreadom monatorious of containers. We begin by introducing the CAM principle, which connects the heavier of some monatories of the distrement. tegen by structuring the CALM projects, which converts the fragment of the summaries in the trans and the darkanets of the MANN and the summaries in the structure of the MANN and the summaries of the summaries of the summaries of the transmission target constraints in distributed Bhots programs. We then do not be support proves, there is the substantiant that structure of the summaries that and the summaries of the summaries of the summaries that and the summaries of the summaries that and the summaries of the summaries that and the summaries of the summaries o

2. CONSISTENCY AND LOGICAL

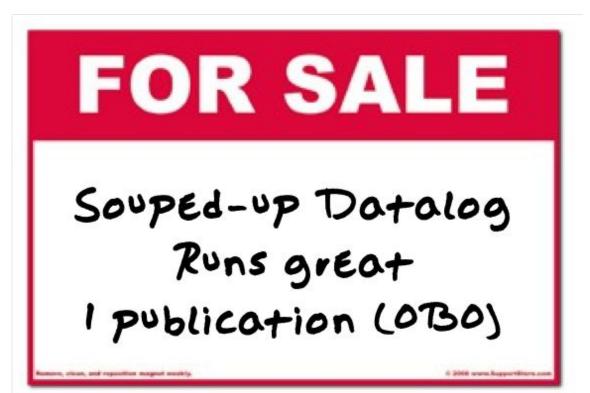
MONOTONICITY (CALM) In this sociator sprease for consocies howevers disributed consistency and logical monotoxisity. This discussion indicates language and analysis local work of sprease howevers and analysis constants hadwork or a longenum in the howevers model. A key problem is dashibited programming in sensoring about the constants the duby of a resolution of an impact and data areas models. Becames delays can be schwarded, analysis and data areas models to became delays can be schwarded, analysis and data areas models. Becames delays can be schwarded, analysis and end and and the model of the schward of an and an areas of the schward of the duby model of the schward of an and an analysis and work and advectures and the schward of the schward of an and a sole in dependence. This article is published under a Centilee Cosmons Attribution Lisener (http://contrivecommons.org/Cosmos/ty/Li/Li, which percents distribution and reprodetory is any medium so well klowing (derivativ socie), pre-ved the pros methods for englished works to the author(s) and CDR 2011, http://www.clu.22.2011.ast/energ.clu.2014/engl.2015.

1. STODUCTION The first weight starting regarding weight shows a starting regarding to a starting regarding shows a starting regarding to a starting regarding shows a starting regarding to a starting regarding shows a starting MONOTONICITY (CALM)

Protocol Repair Using Lineage Graphs

Lemmet Oldenburg<sup>1,2</sup>, Xingforg Zhu<sup>1</sup>, Kamala Ramasabramanian<sup>1</sup>, and Peter Abuso<sup>1</sup>

### Poor lucky me



#### BTW: Where's the lie?