ALVARO VIDELA - @old_sound

METAPHORS WE COMPUTE BY

THE YEAR IS 1980

GEORGE LAKOFF & MARK JOHNSON

METAPHORS WE LIVE BY

METAPHOR ISN'T JUST A MATTER OF POETRY AND RHETORICAL FLOURISH

METAPHORS DICTATE

METAPHORS DICTATE

- How we think

METAPHORS DICTATE

- How we think
- How we behave

METAPHORS DICTATE

- How we think
- How we behave
- How we perceive

METAPHORS DICTATE

- How we think
- How we behave
- How we perceive
- How our conceptual system is built

ARGUMENT IS WAR

ARGUMENT IS WAR

ARGUMENT IS WAR

- Your claims are ***indefensible***

ARGUMENT IS WAR

- Your claims are ***indefensible***
- He ***attacked every weak point*** in my argument

ARGUMENT IS WAR

- Your claims are ***indefensible***
- He ***attacked every weak point*** in my argument
- I ***demolished*** his argument

ARGUMENT IS WAR

- Your claims are ***indefensible***
- He ***attacked every weak point*** in my argument
- I ***demolished*** his argument
- I never ***won*** an argument with him

ARGUMENT IS WAR

- Your claims are ***indefensible***
- He ***attacked every weak point*** in my argument
- I ***demolished*** his argument
- I never ***won*** an argument with him
- His criticisms were ***right on target***

WHAT IF ARGUMENT IS A DANCE?

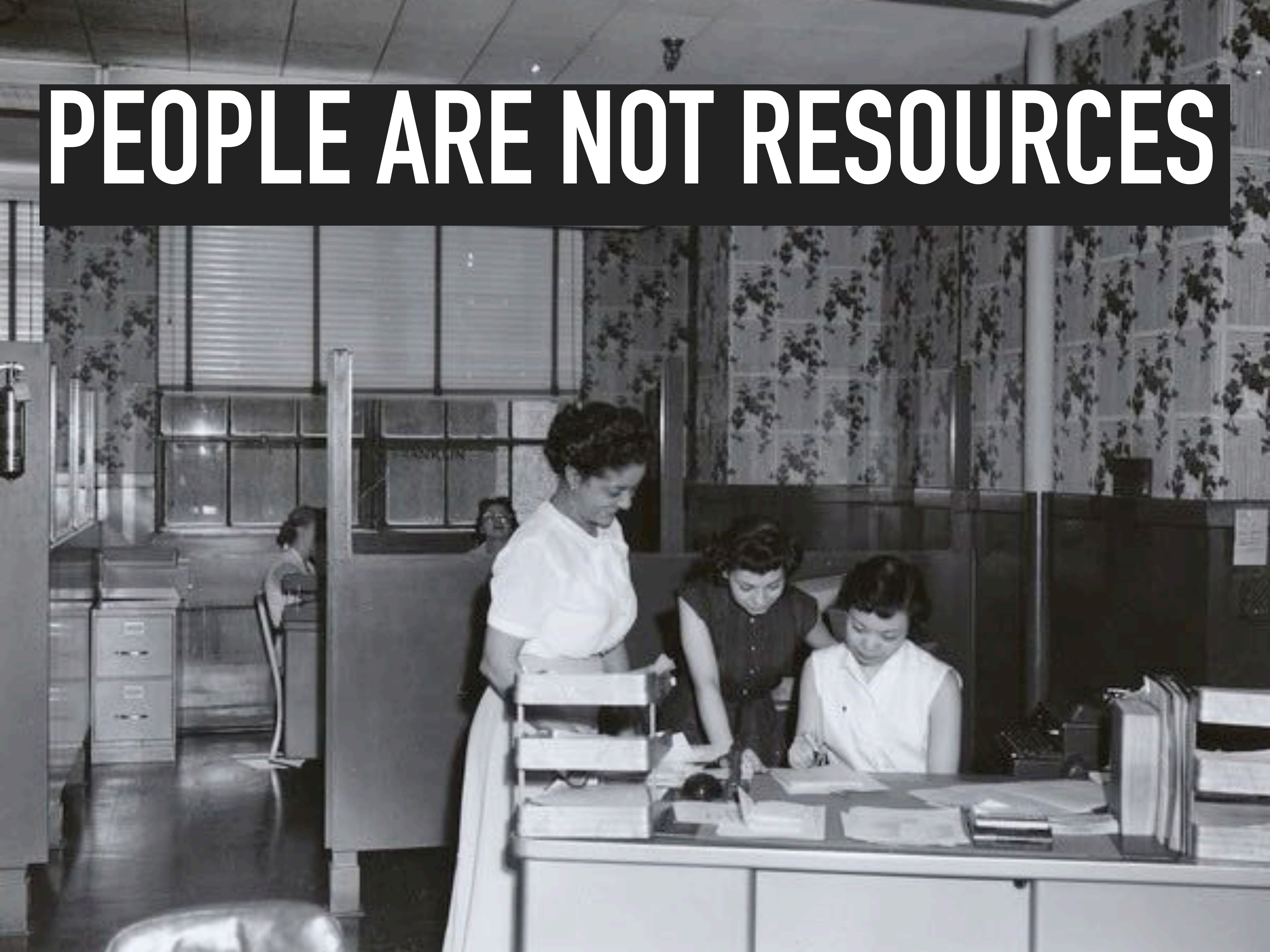
I'M NOT CONVINCED

LET'S TALK ABOUT POLITICS

I’M STILL NOT CONVINCED

HUMAN RESOURCE MANAGEMENT

PEOPLE ARE NOT RESOURCES



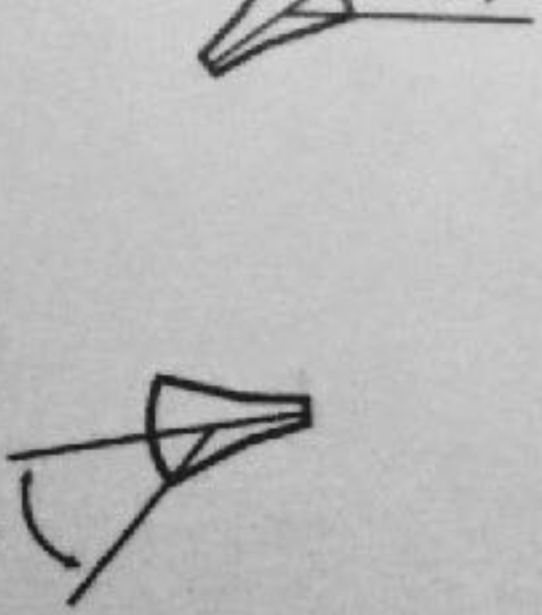
TRIGGER WARNING

GIVING A PLATFORM TO RACISTS

“WRESTLING WITH INCLUSION AT XYZCONF”

“WRESTLING WITH INCLUSION AT XYZCONF”

LET'S TALK ABOUT COMPUTERS



MARGOT LEE SHETTERLY

HIDDEN

COMPUTERS

METAPHORS ENABLE UNDERSTANDING

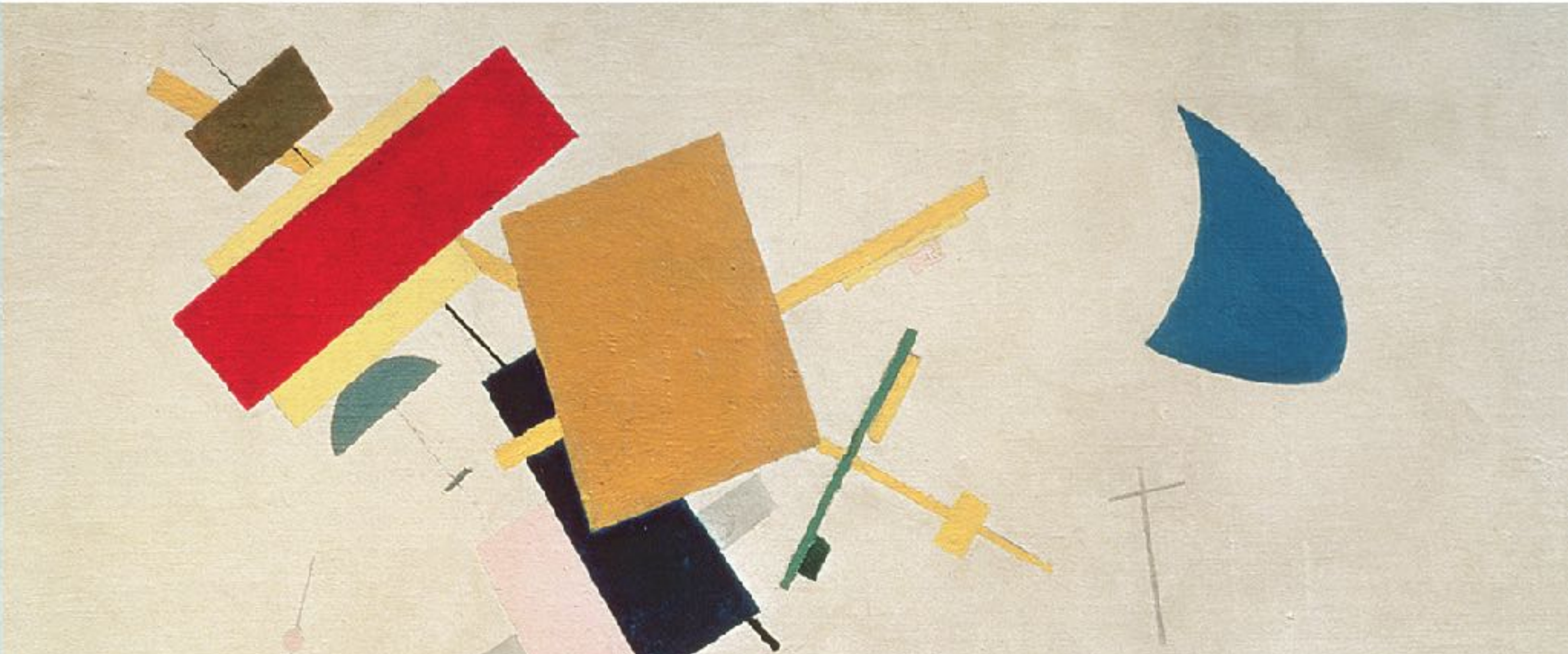
JULIET IS LIKE THE SUN

JULIET GAVE ME SKIN CANCER

THE GEOMETRY OF MEANING

SEMANTICS BASED ON CONCEPTUAL SPACES

PETER GÄRDENFORS



METAPHORICAL MAPPINGS PRESERVE THE THE COGNITIVE TOPOLOGY OF THE SOURCE DOMAIN

IN A WAY CONSISTENT WITH THE INHERENT STRUCTURE OF THE TARGET DOMAIN

METAPHORS TRANSFER INFORMATION FROM ONE CONCEPTUAL DOMAIN TO ANOTHER

WHAT IS TRANSFERRED IS A PATTERN RATHER THAN DOMAIN SPECIFIC INFORMATION

A METAPHOR CAN THUS BE USED TO IDENTIFY A STRUCTURE IN A DOMAIN THAT WOULD NOT HAVE BEEN DISCOVERED OTHERWISE

THIS IS HOW METAPHORS CREATE NEW KNOWLEDGE

METAPHORS OBSCURE UNDERSTANDING

TELE-GRAPH

"SOMETIMES OUR TOOLS DO WHAT WE TELL THEM TO. OTHER TIMES, WE ADAPT OURSELVES TO OUR TOOLS' REQUIREMENTS"

Nicholas Carr

METAPHORS ARE THE TOOLS OF THOUGHT

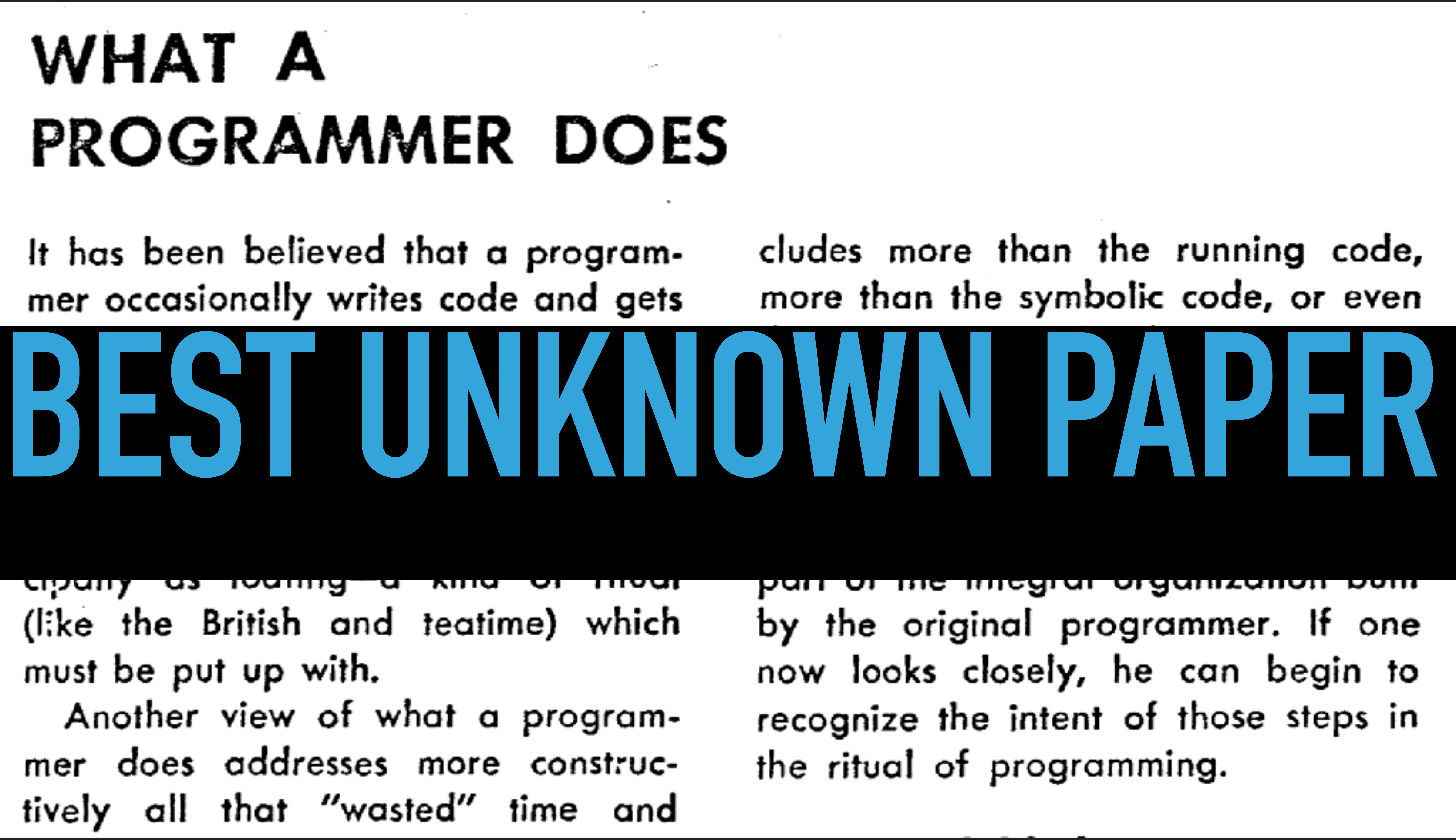
METAPHORS AND CODE

WHAT A PROGRAMMER DOES

It has been believed that a programmer occasionally writes code and gets it running on a computer, and that this is what he is paid for. In spite of his obvious inefficiency, no one else seems to do this work more effectively. However, his activity is still observed principally as loafing—a kind of ritual (like the British and teatime) which must be put up with.

Another view of what a programmer does addresses more constructively all that "wasted" time and includes more than the running code, more than the symbolic code, or even the operator's guide, the maintenance guide, or the design guide. For in fact, in response to any serious breach of the program's integrity, a programmer will become involved, as part of the integral organization built by the original programmer. If one now looks closely, he can begin to recognize the intent of those steps in the ritual of programming.

BEST UNKNOWN PAPER



WHAT A PROGRAMMER DOES

It has been believed that a programmer occasionally writes code and gets

[redacted]

(like the British and teatime) which must be put up with.

Another view of what a programmer does addresses more constructively all that "wasted" time and

cludes more than the running code, more than the symbolic code, or even

[redacted]

by the original programmer. If one now looks closely, he can begin to recognize the intent of those steps in the ritual of programming.

A PROGRAMER'S ROLE

“TO PROGRAM IS TO WRITE TO ANOTHER PROGRAMMER ABOUT OUR SOLUTION TO A PROBLEM”

What a Programmer Does

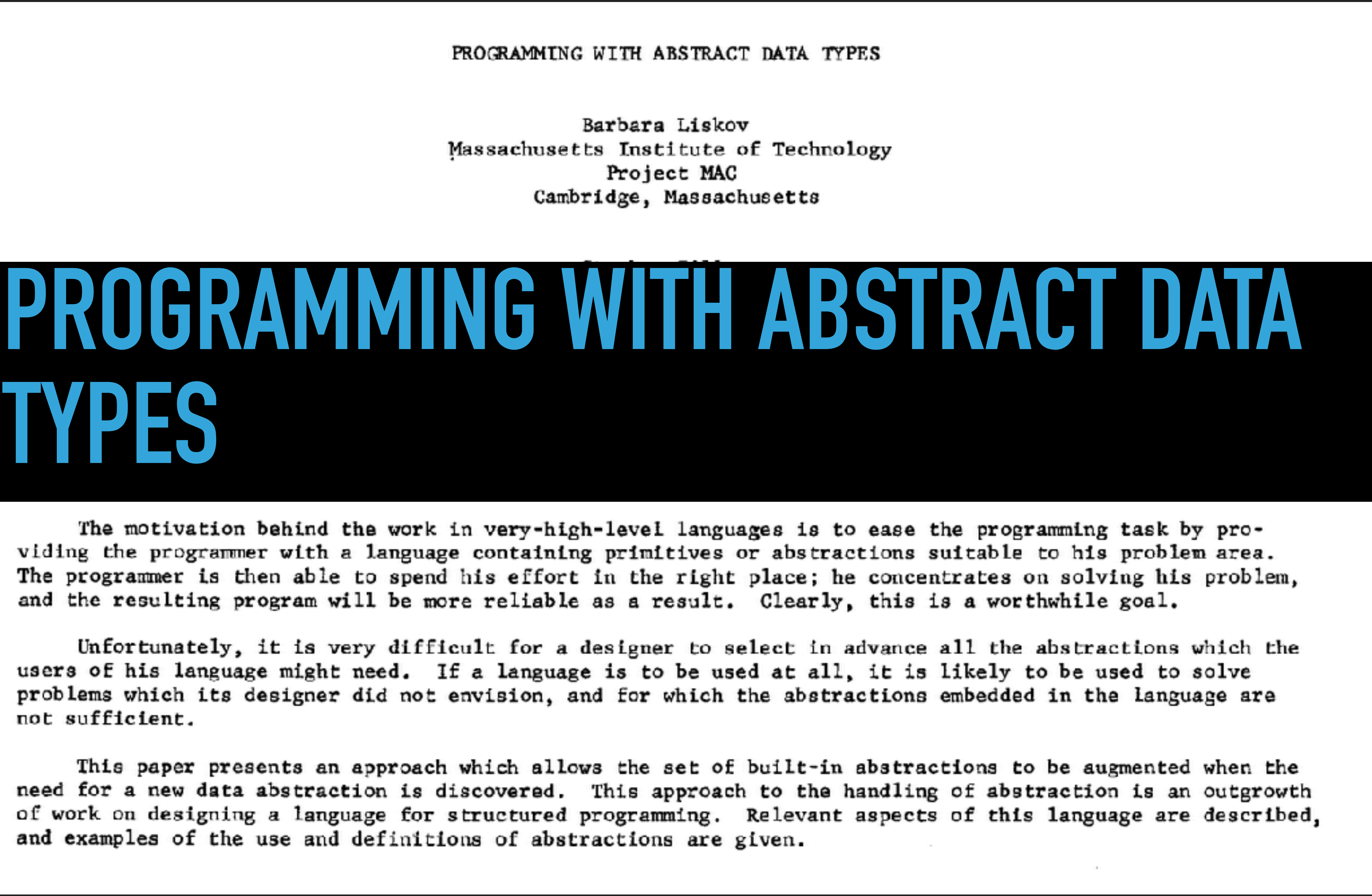
A PROGRAMER'S ROLE

"NO ONE HAS SEEN A PROGRAM WHICH THE MACHINE COULD NOT COMPREHEND BUT WHICH HUMANS DID"

What a Programmer Does

TYPES ARE THE CHARACTERS THAT TELL THE STORY OF OUR PROGRAMS

PROGRAMMING WITH ABSTRACT DATA TYPES



PROGRAMMING WITH ABSTRACT DATA TYPES

Barbara Liskov
Massachusetts Institute of Technology
Project MAC
Cambridge, Massachusetts

The motivation behind the work in very-high-level languages is to ease the programming task by providing the programmer with a language containing primitives or abstractions suitable to his problem area. The programmer is then able to spend his effort in the right place; he concentrates on solving his problem, and the resulting program will be more reliable as a result. Clearly, this is a worthwhile goal.

Unfortunately, it is very difficult for a designer to select in advance all the abstractions which the users of his language might need. If a language is to be used at all, it is likely to be used to solve problems which its designer did not envision, and for which the abstractions embedded in the language are not sufficient.

This paper presents an approach which allows the set of built-in abstractions to be augmented when the need for a new data abstraction is discovered. This approach to the handling of abstraction is an outgrowth of work on designing a language for structured programming. Relevant aspects of this language are described, and examples of the use and definitions of abstractions are given.

WITHOUT TYPES WE JUST HAVE OPERATIONS ON STREAM OF BYTES

CHOOSING THE RIGHT DATA STRUCTURE

CHOOSE THE RIGHT DATA STRUCTURE

CHOOSE THE RIGHT DATA STRUCTURE

- Array

CHOOSE THE RIGHT DATA STRUCTURE

- Array
- Set

CHOOSE THE RIGHT DATA STRUCTURE

- Array
- Set
- LinkedList

CHOOSE THE RIGHT DATA STRUCTURE

- Array
- Set
- LinkedList
- Queue

CHOOSE THE RIGHT DATA STRUCTURE

- Array
- Set
- LinkedList
- Queue
- Stack

A PROGRAM'S EXPLANATORY POWER IS THE MEASURE OF ITS OWN ELEGANCE

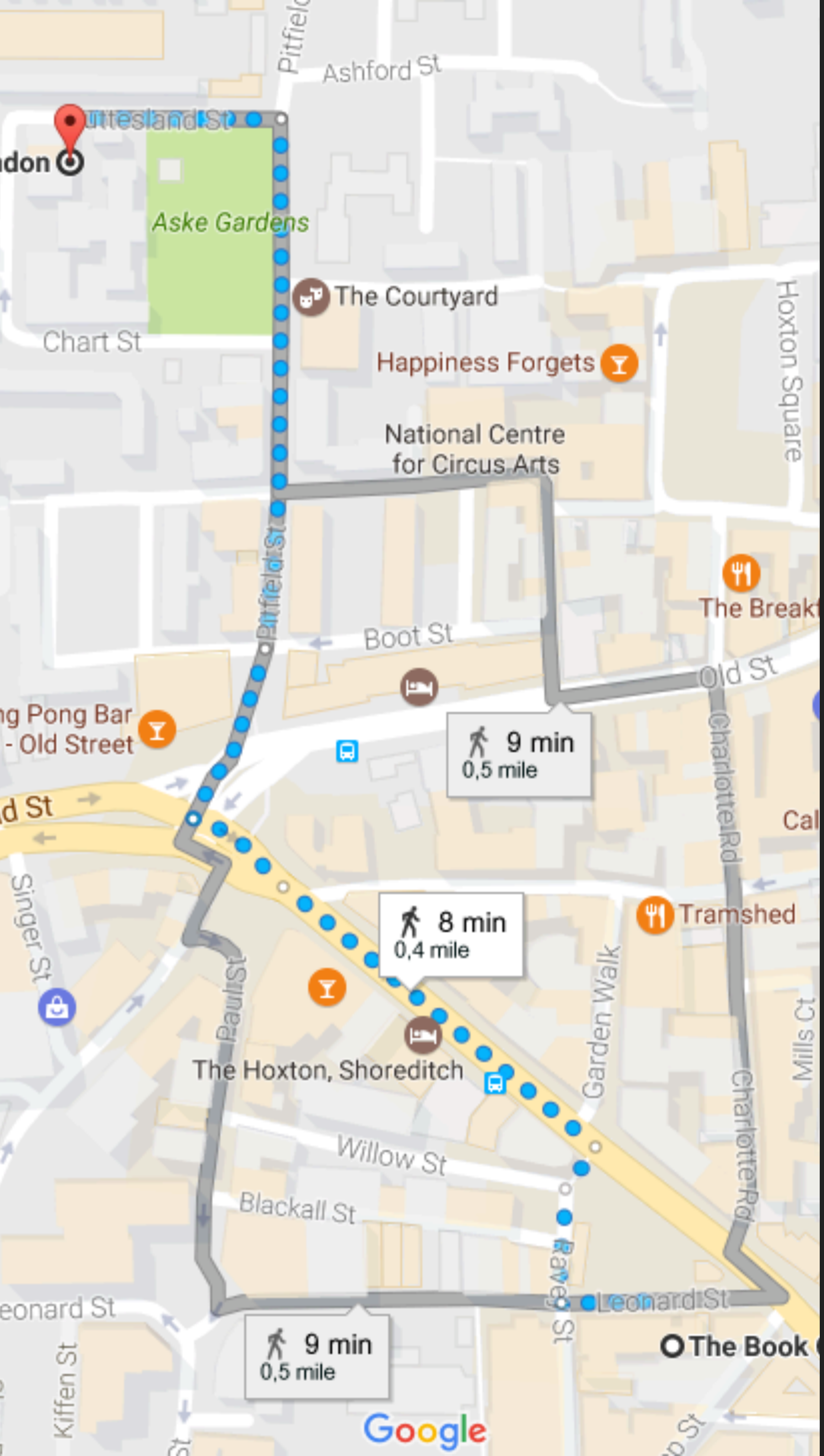
DATA STRUCTURES HAVE EXPLANATORY POWER

COGNITIVE LEAPS

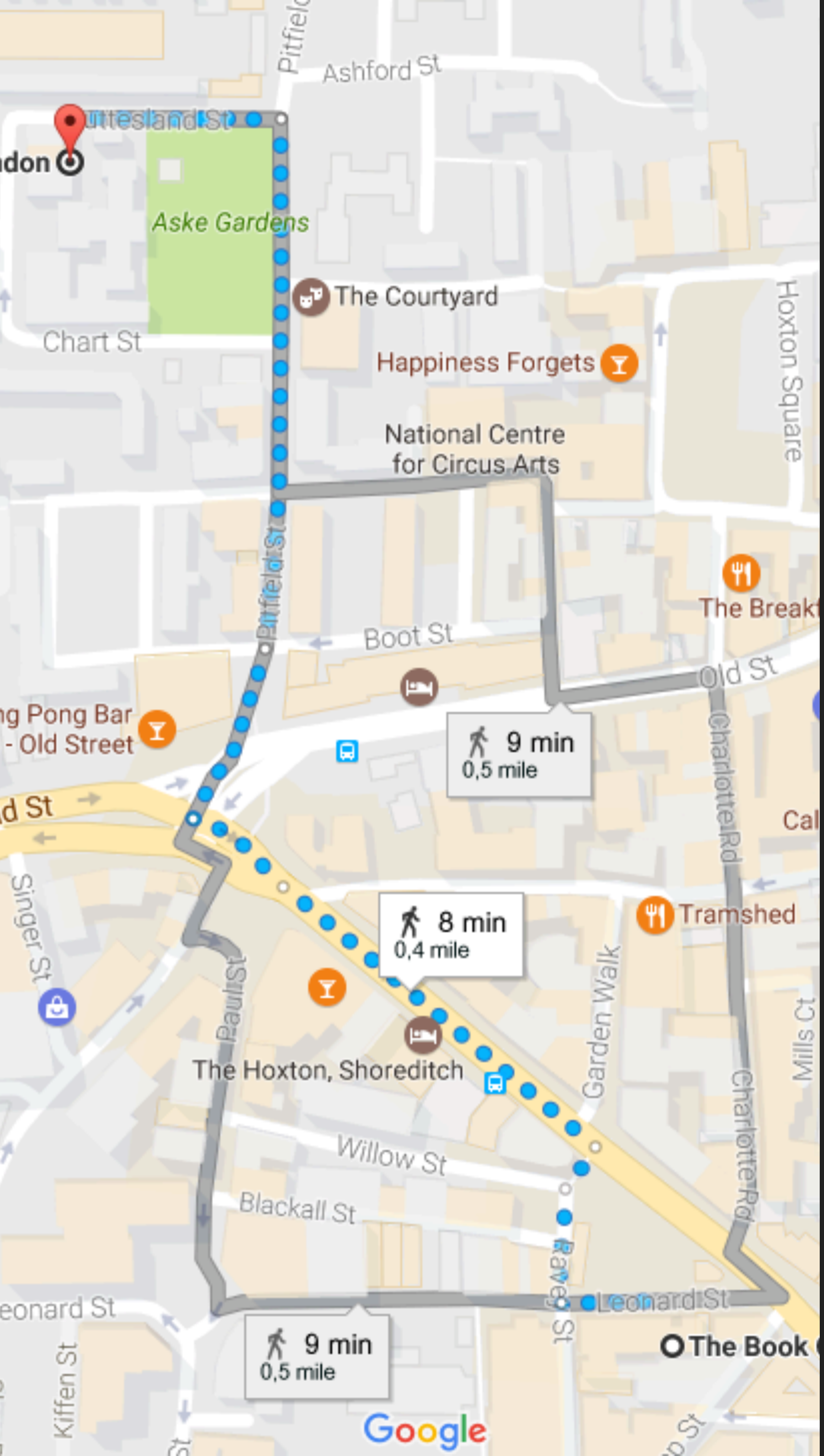
TASK SCHEDULING

TASK SCHEDULING

QUEUEING THEORY



ROUTE PLANNING



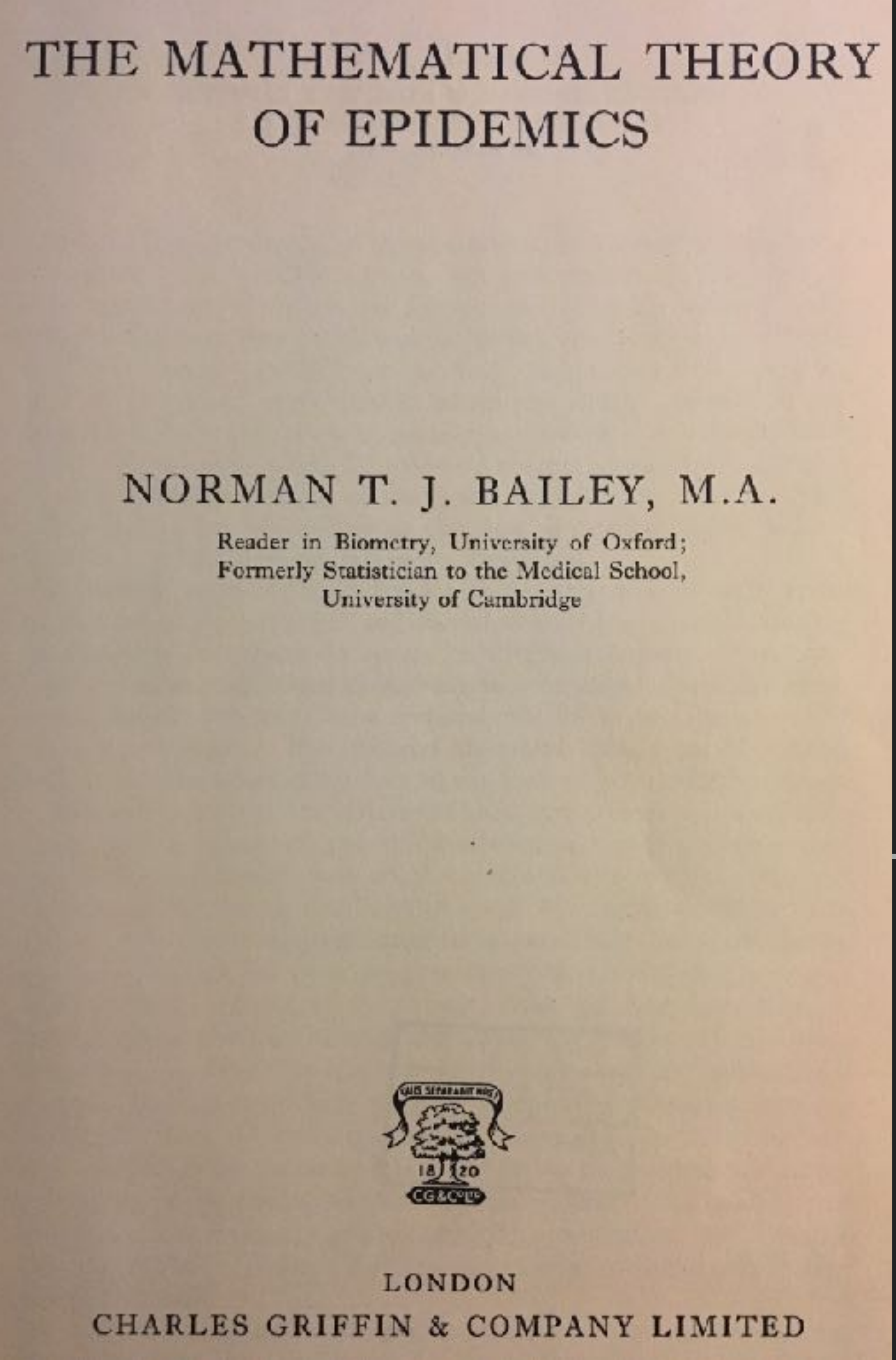
ROUTE PLANNING

GRAPH THEORY

DATABASE REPLICATION

DATABASE REPLICATION

RUMOUR MONGERING



THE MATHEMATICAL THEORY OF EPIDEMICS

NORMAN T. J. BAILEY, M.A.

Reader in Biometry, University of Oxford;
Formerly Statistician to the Medical School,
University of Cambridge

LONDON
CHARLES GRIFFIN & COMPANY LIMITED

DATABASE REPLICATION

EPIDEMICS

SO EVERYTHING IS A METAPHOR?

I DON'T BELIEVE YOU

DISTRIBUTED SYSTEMS METAPHORS

Whenever ***nodes*** need to ***agree*** on a common value, we start a ***consensus*** algorithm to ***decide*** on a value. There's usually a ***leader*** process that takes care of making the final decision based on the ***votes*** it has received from its ***peers***.

DISTRIBUTED SYSTEMS METAPHORS

Nodes communicate sending ***messages*** over a ***channel***, which might get ***congested*** due to ***too much traffic***. This could create an information ***bottleneck***, with ***queues*** at each end of the ***channels*** backing up.

DISTRIBUTED SYSTEMS METAPHORS

These ***bottlenecks*** might render one or more nodes **unresponsive**, causing ***network partitions***. Is the process that's taking too long to ***respond dead***? We won't know unless we set a timeout…

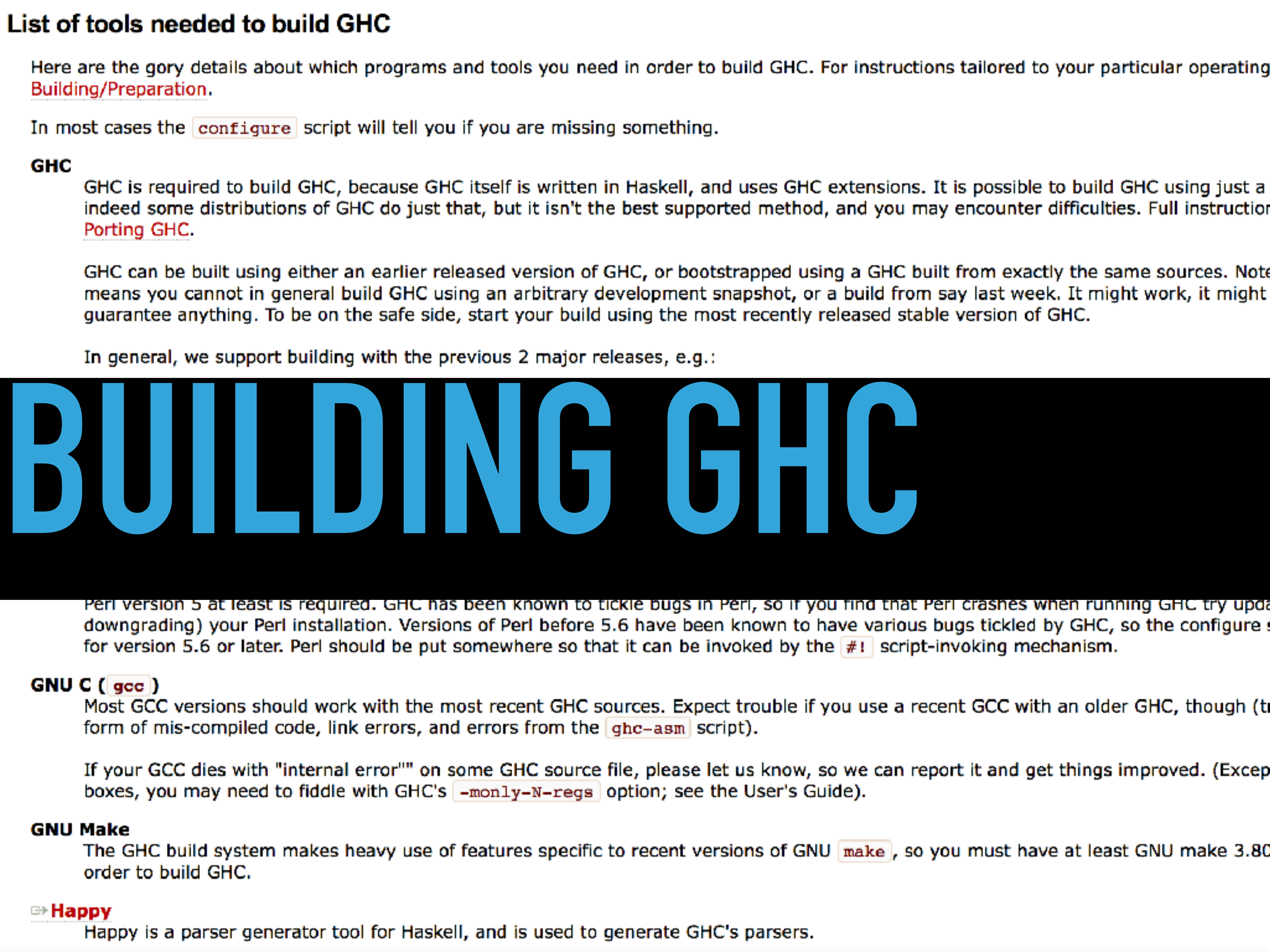
INTERMEZZO

PACKAGING

10 year
General Purpose

cut tab to open

BUILDING GHC



List of tools needed to build GHC

Here are the gory details about which programs and tools you need in order to build GHC. For instructions tailored to your particular operating
Building/Preparation.

In most cases the `configure` script will tell you if you are missing something.

GHC

GHC is required to build GHC, because GHC itself is written in Haskell, and uses GHC extensions. It is possible to build GHC using just a
indeed some distributions of GHC do just that, but it isn't the best supported method, and you may encounter difficulties. Full instructio
Porting GHC.

GHC can be built using either an earlier released version of GHC, or bootstrapped using a GHC built from exactly the same sources. Note
means you cannot in general build GHC using an arbitrary development snapshot, or a build from say last week. It might work, it might
guarantee anything. To be on the safe side, start your build using the most recently released stable version of GHC.

In general, we support building with the previous 2 major releases, e.g.:

Perl version 5 at least is required. GHC has been known to tickle bugs in Perl, so if you find that Perl crashes when running GHC try upda
downgrading) your Perl installation. Versions of Perl before 5.6 have been known to have various bugs tickled by GHC, so the configure s
for version 5.6 or later. Perl should be put somewhere so that it can be invoked by the `#!` script-invoking mechanism.

GNU C (`gcc`)

Most GCC versions should work with the most recent GHC sources. Expect trouble if you use a recent GCC with an older GHC, though (t
form of mis-compiled code, link errors, and errors from the `ghc-asm` script).

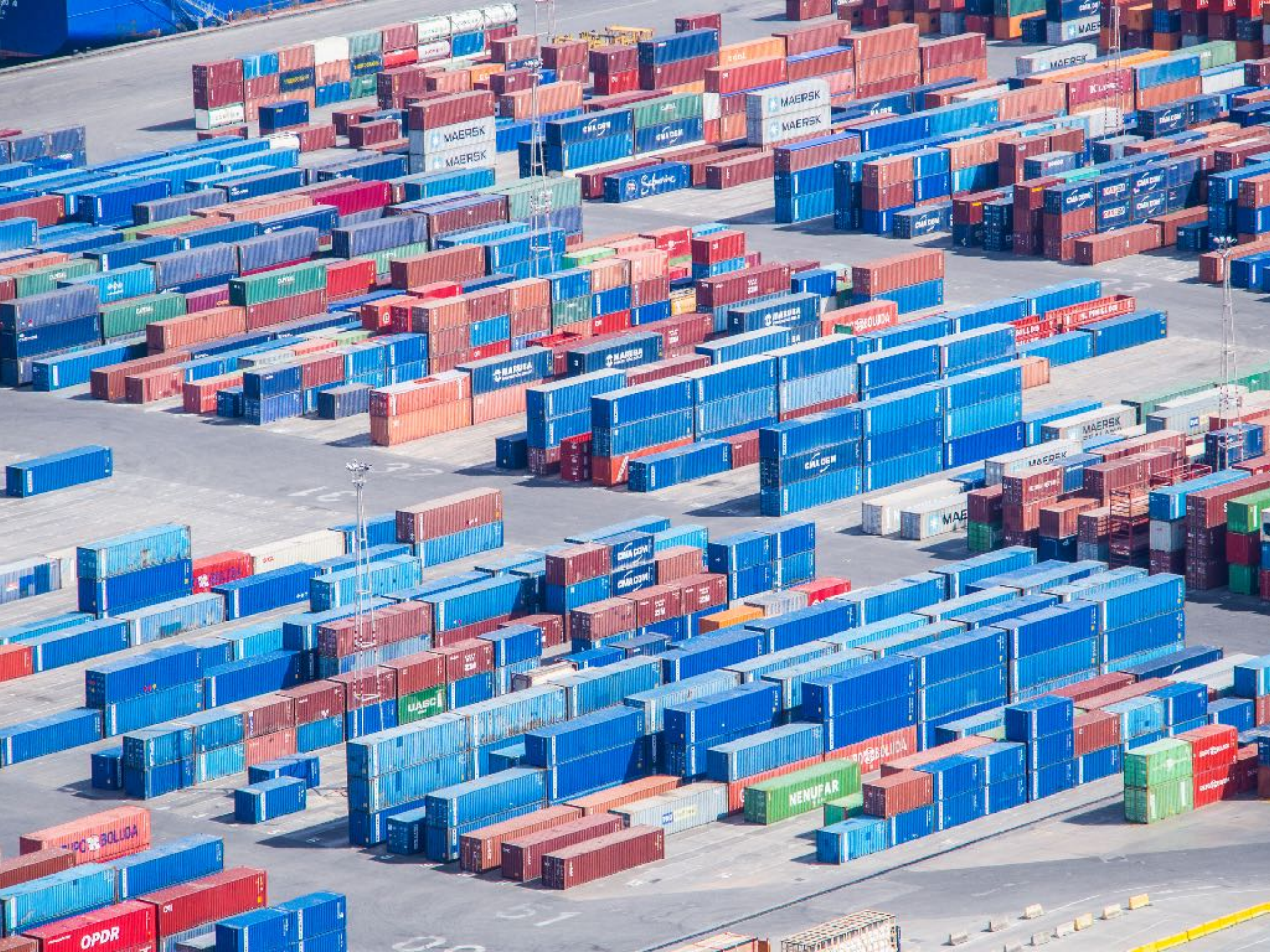
If your GCC dies with "internal error"" on some GHC source file, please let us know, so we can report it and get things improved. (Excep
boxes, you may need to fiddle with GHC's `-monly-N-regs` option; see the User's Guide).

GNU Make

The GHC build system makes heavy use of features specific to recent versions of GNU `make`, so you must have at least GNU make 3.80
order to build GHC.

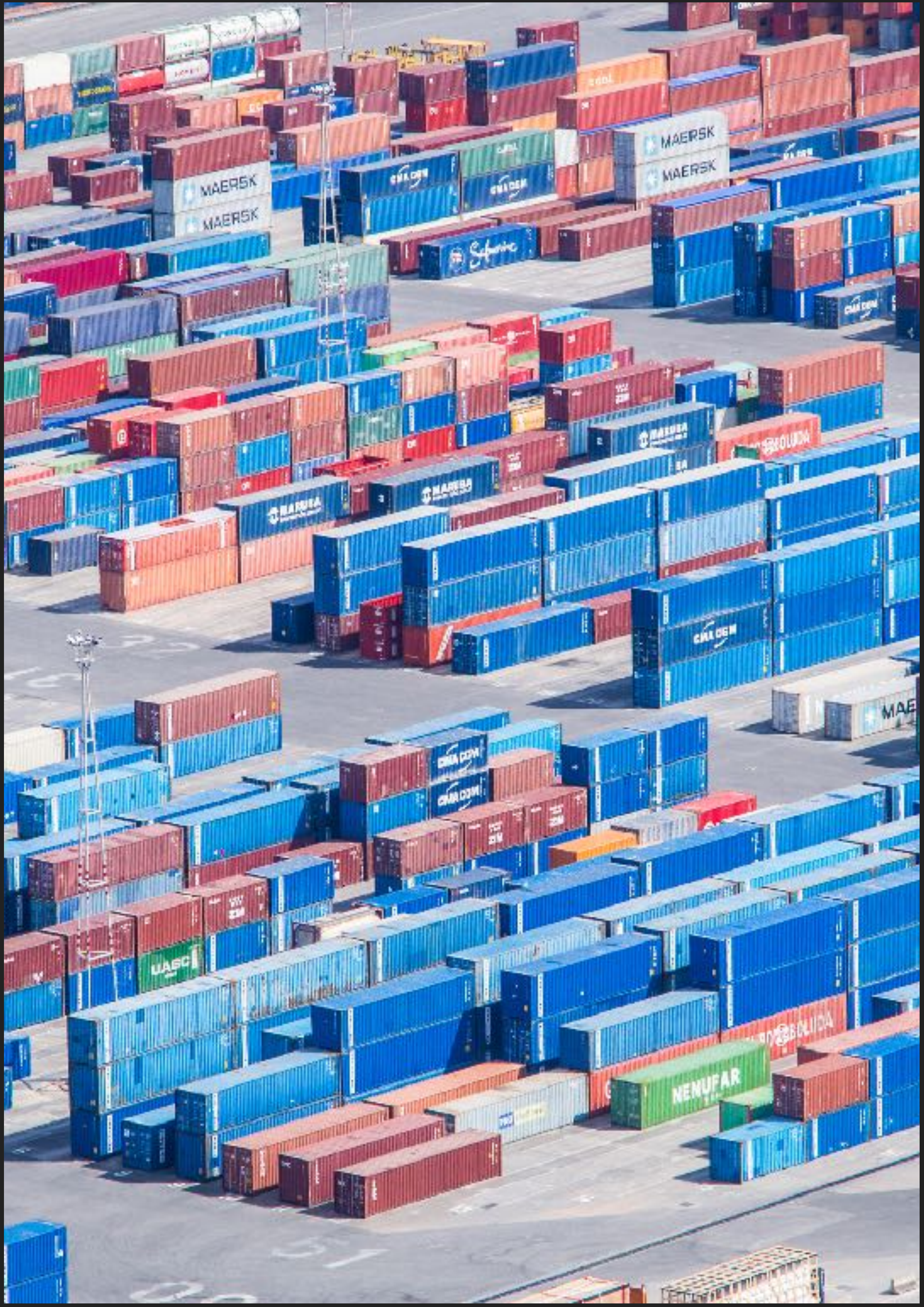
Happy

Happy is a parser generator tool for Haskell, and is used to generate GHC's parsers.

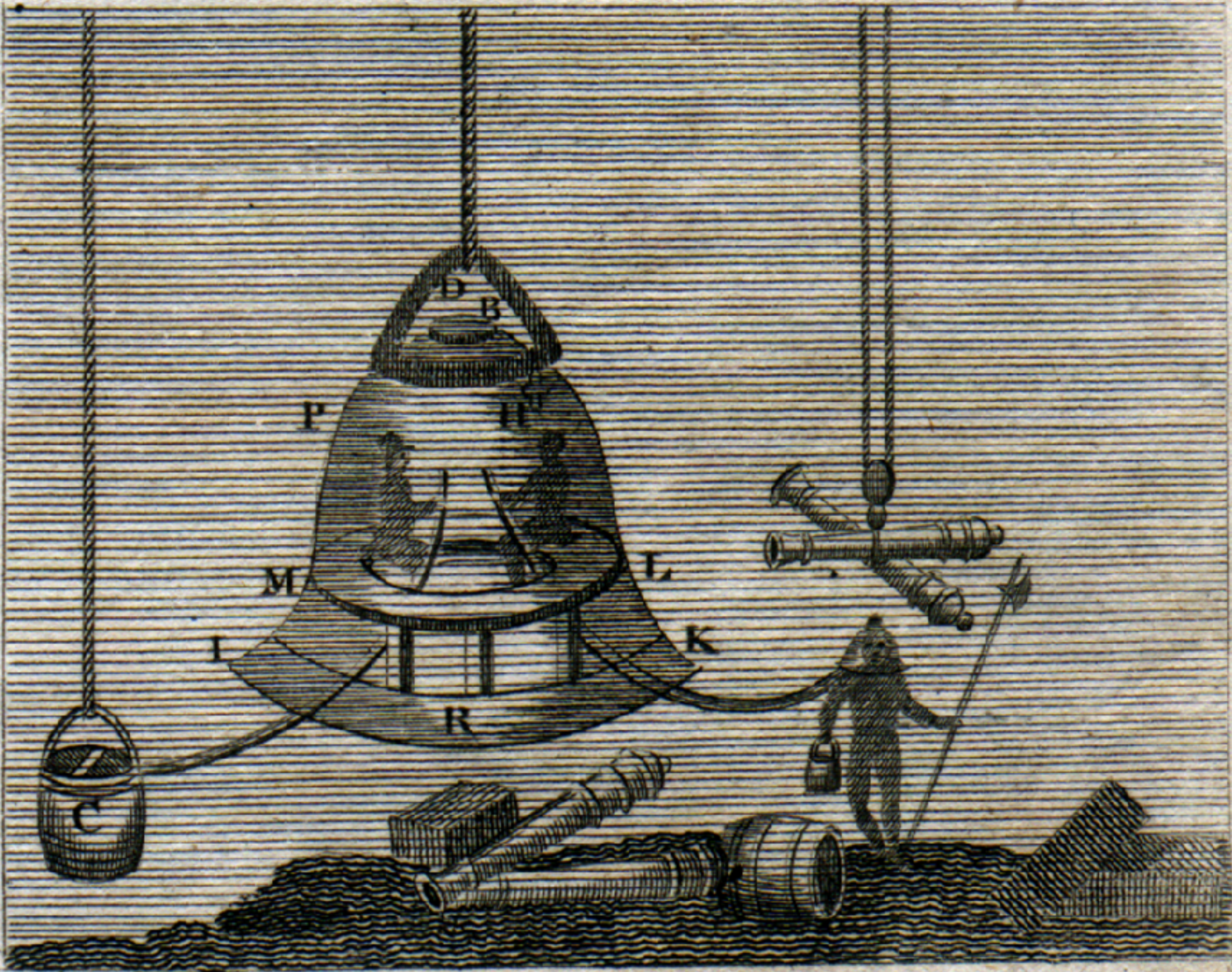


CONTAINERS

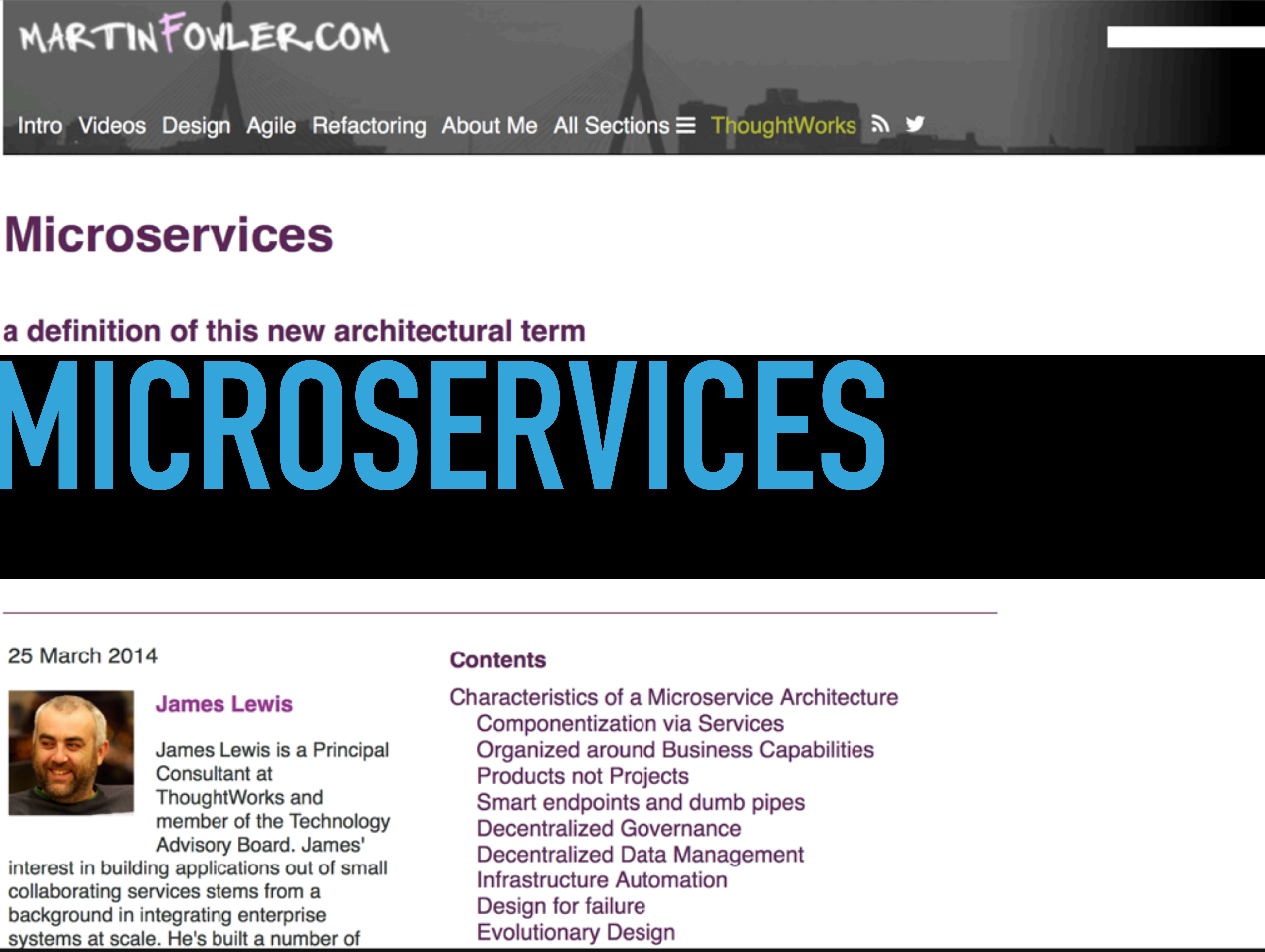
- Standard
- Ship Anywhere
- Train, Ships, Trucks
- Stackable
- Reusable



Halley's Diving Bell.



MICROSERVICES



MARTINFOWLER.COM

Intro Videos Design Agile Refactoring About Me All Sections ThoughtWorks

Microservices

a definition of this new architectural term

25 March 2014

James Lewis

James Lewis is a Principal Consultant at ThoughtWorks and member of the Technology Advisory Board. James' interest in building applications out of small collaborating services stems from a background in integrating enterprise systems at scale. He's built a number of

Contents

MICROSERVICES

- Decentralised Governance
- Monolith vs. Microservice
- Isolation
- Collaboration
- Small is better - Big is cumbersome
- David vs. Goliath

BRING POWER BACK TO THE DEVELOPER AND THE DEVELOPER WILL MAKE YOU A KING

ERLANG ANYONE?

“IN ANOTHER DIRECTION, ONE COULD ARGUE THAT MICROSERVICES ARE THE SAME THING AS THE ERLANG PROGRAMMING MODEL, BUT APPLIED TO AN ENTERPRISE APPLICATION CONTEXT”

WHAT'S ERLANG'S ELEVATOR PITCH?

MASTER THE ART OF METAPHOR SELECTION

FIRST GET PEOPLE TO UNDERSTAND THINGS

THEN EXPLAIN HOW THINGS ACTUALLY WORK

RABBITMQ
A JOB SERVER?

MASTER THE ART OF MEANING AMPLIFICATION

OUR PROGRAM IS THE METAPHOR FOR THE SOLUTION WE FOUND

REFERENCES

- Lakoff, George, and Mark Johnson. “Metaphors We Live By”
- Gärdenfors, Peter. “The Geometry of Meaning”
- Gleick, James. “The Information: A History, a Theory, a Flood”
- Geary, James. “I Is an Other: The Secret Life of Metaphor and How It Shapes the Way We See the World”
- Demers, Alan, Dan Greene, Carl Hauser, Wes Irish, and John Larson. “Epidemic Algorithms for Replicated Database Maintenance”

CREDITS - CC BY-NC-ND

- Office Workers: https://flic.kr/p/5WwpeV
- Sun: https://flic.kr/p/9Q6SY1
- Queue: https://flic.kr/p/8AqWW7
- Consensus: https://flic.kr/p/aws7dH
- Bottlenecks: https://flic.kr/p/EJ5Q3
- Gossip: https://flic.kr/p/4bCDr2
- Containers: https://flic.kr/p/nWLQxE

THANK YOU!

@old_sound