Introduction to Red

From system programming to scripting

Introducing myself

- Nenad Rakocevic, 40, french
- Programming since 25 years:
  - in C/C++, *Basic, ASM, REBOL, web client-side languages,…
  - was a happy Amiga user and registered BeOS developer
- Founder of two software companies in Paris:
  - Softinnov
  - ElasticSoft
- Author of several libraries for REBOL:
  - MySQL, PostgresQL, LDAP native drivers
  - UniServe: asynchronous, event-driven network engine
  - Cheyenne Web Server: full-featured web application server
  - CureCode: very fast web-based bug tracker (Mantis-like)

Why make yet another language?

To build an efficient new tool.

To have an open source implementation of REBOL language.

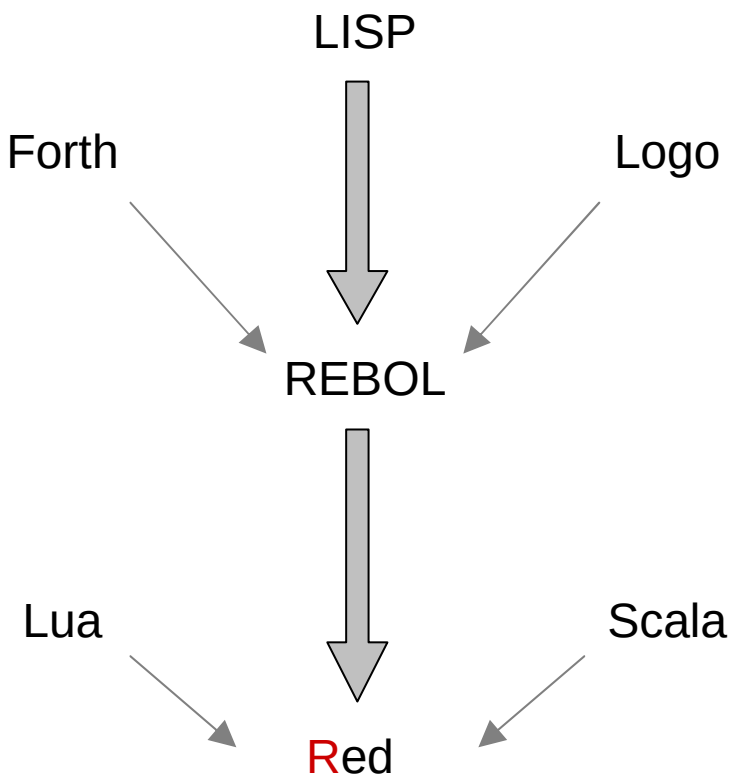
It is a very exciting challenge.

What is REBOL?

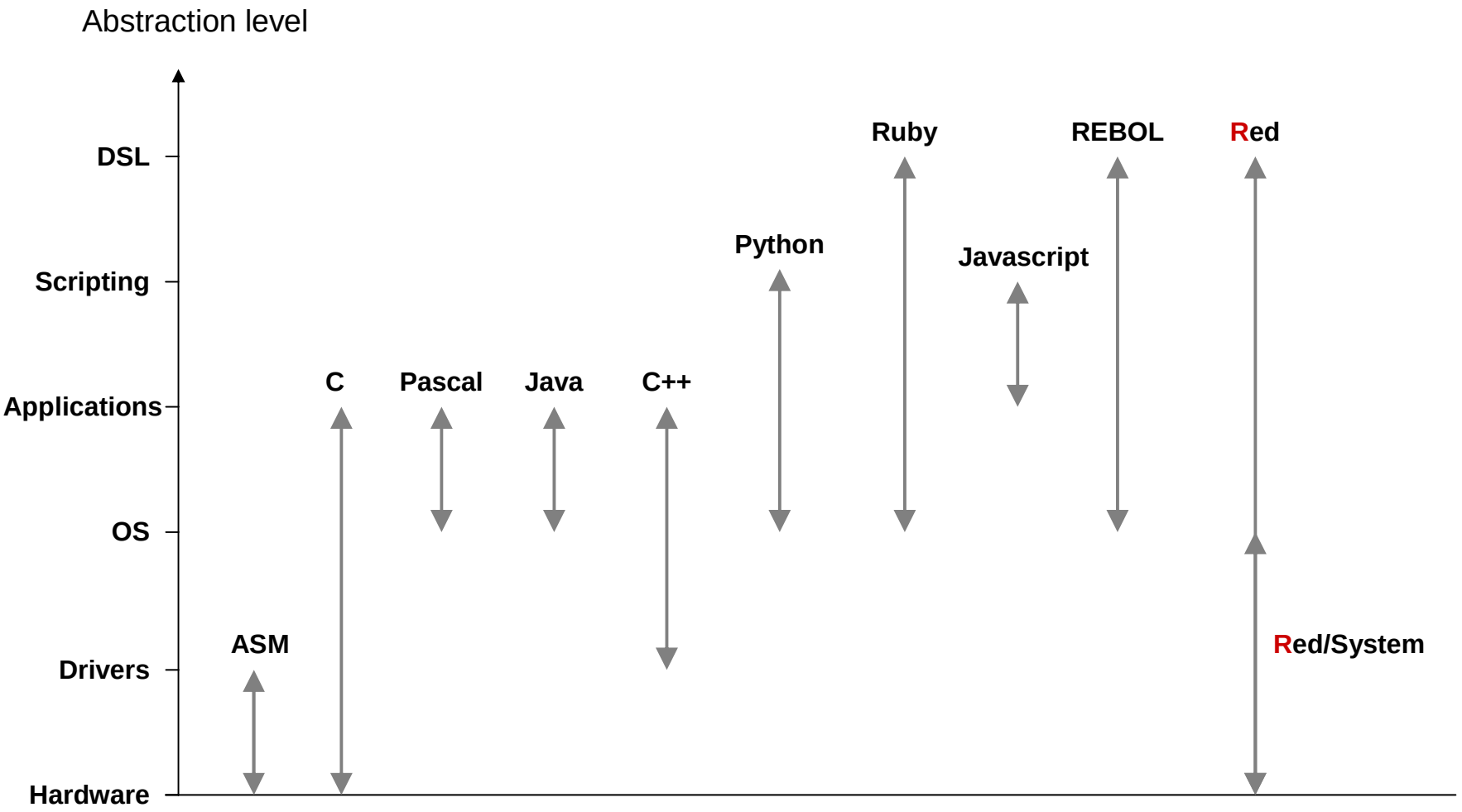
- Invented by Carl Sassenrath (AmigaOS kernel father)
- Available since 1998
- Abandoned since a year by its author
- Closed source (part of standard library has been opened)

- Interpreted
- Multi-paradigm (imperative, functional, OO, declarative)
- Strong meta-programming abilities

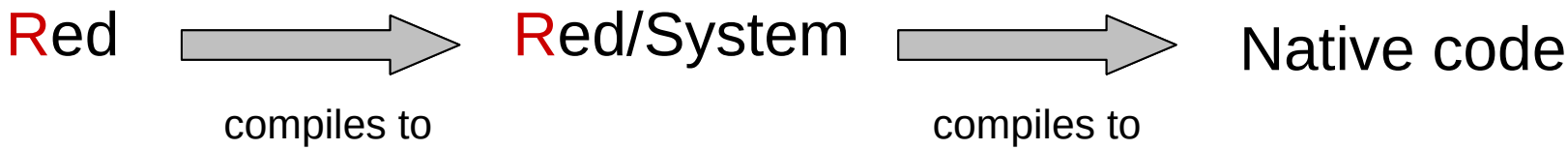
Red quick tour: Genealogy



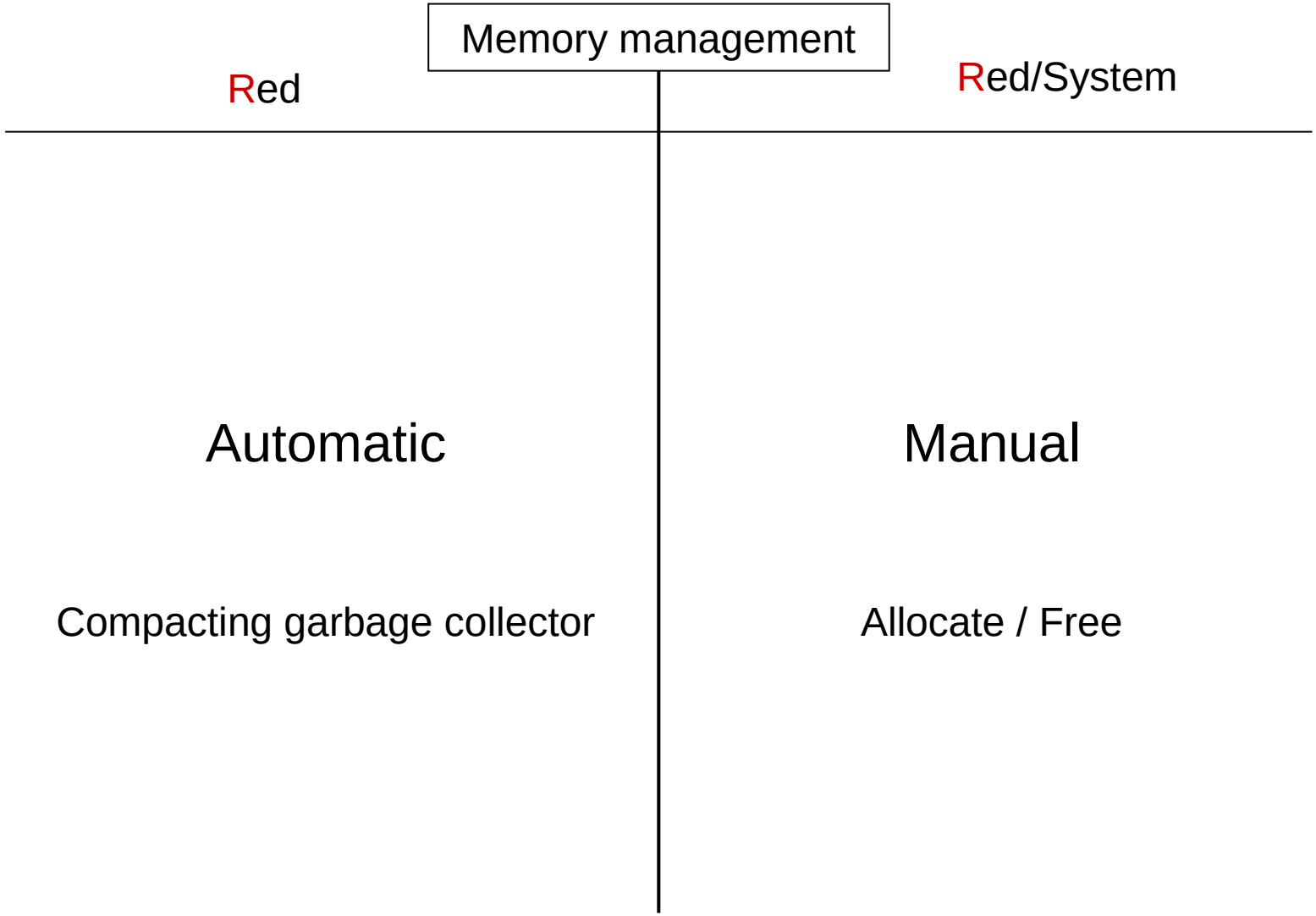
Red quick tour: Natural range of application



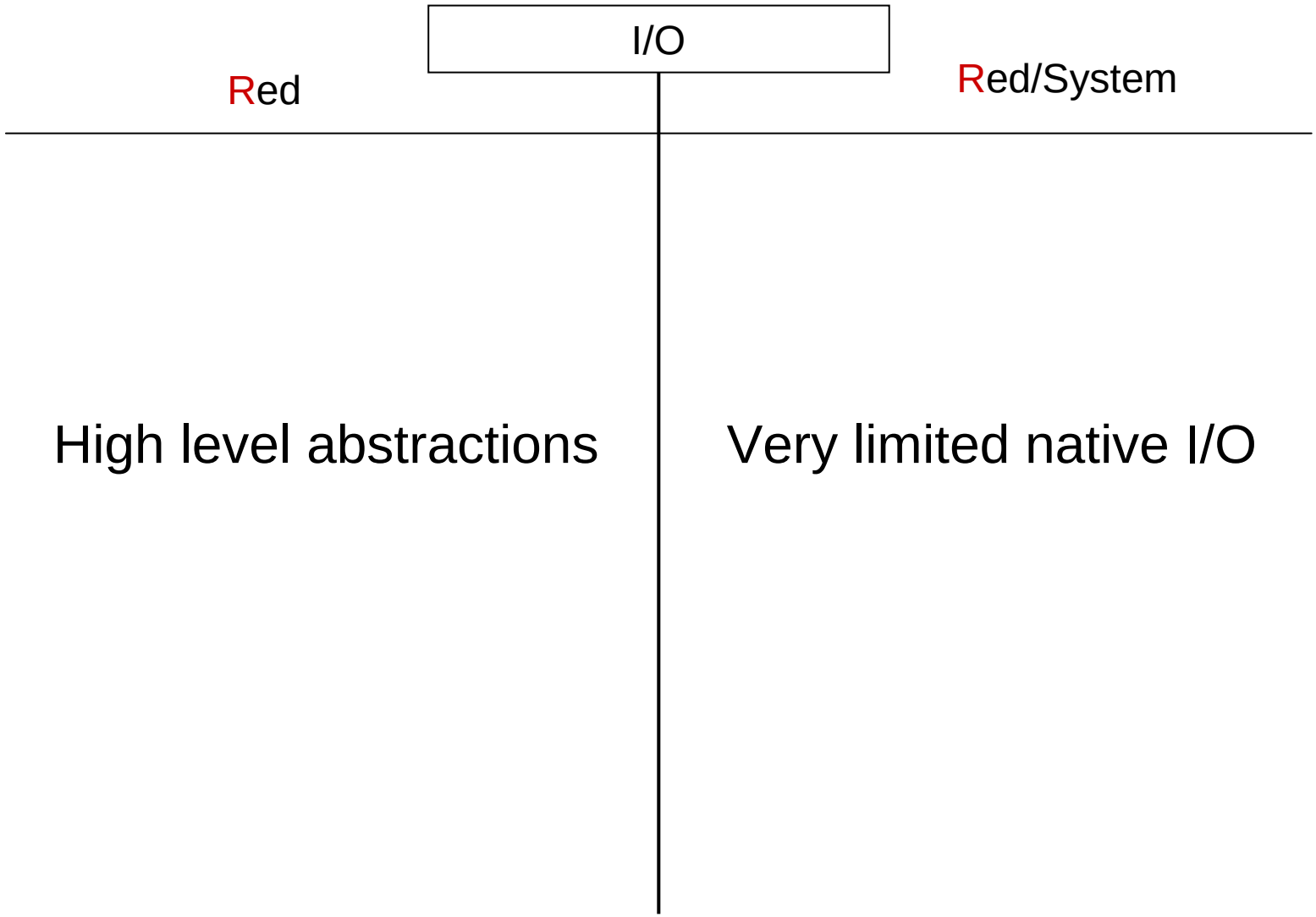
Red quick tour: Language stack



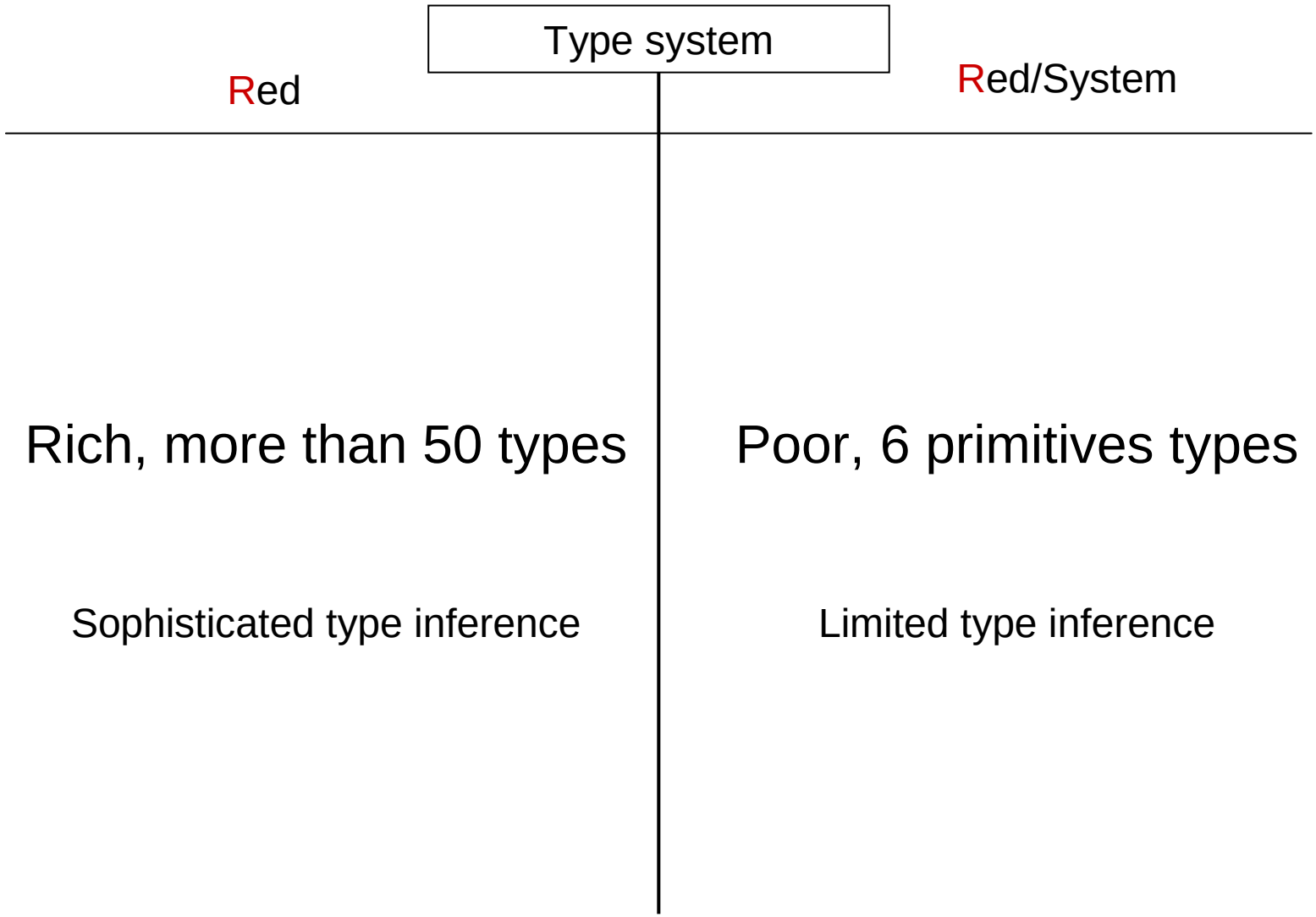
Red quick tour: Red vs Red/System (1/6)



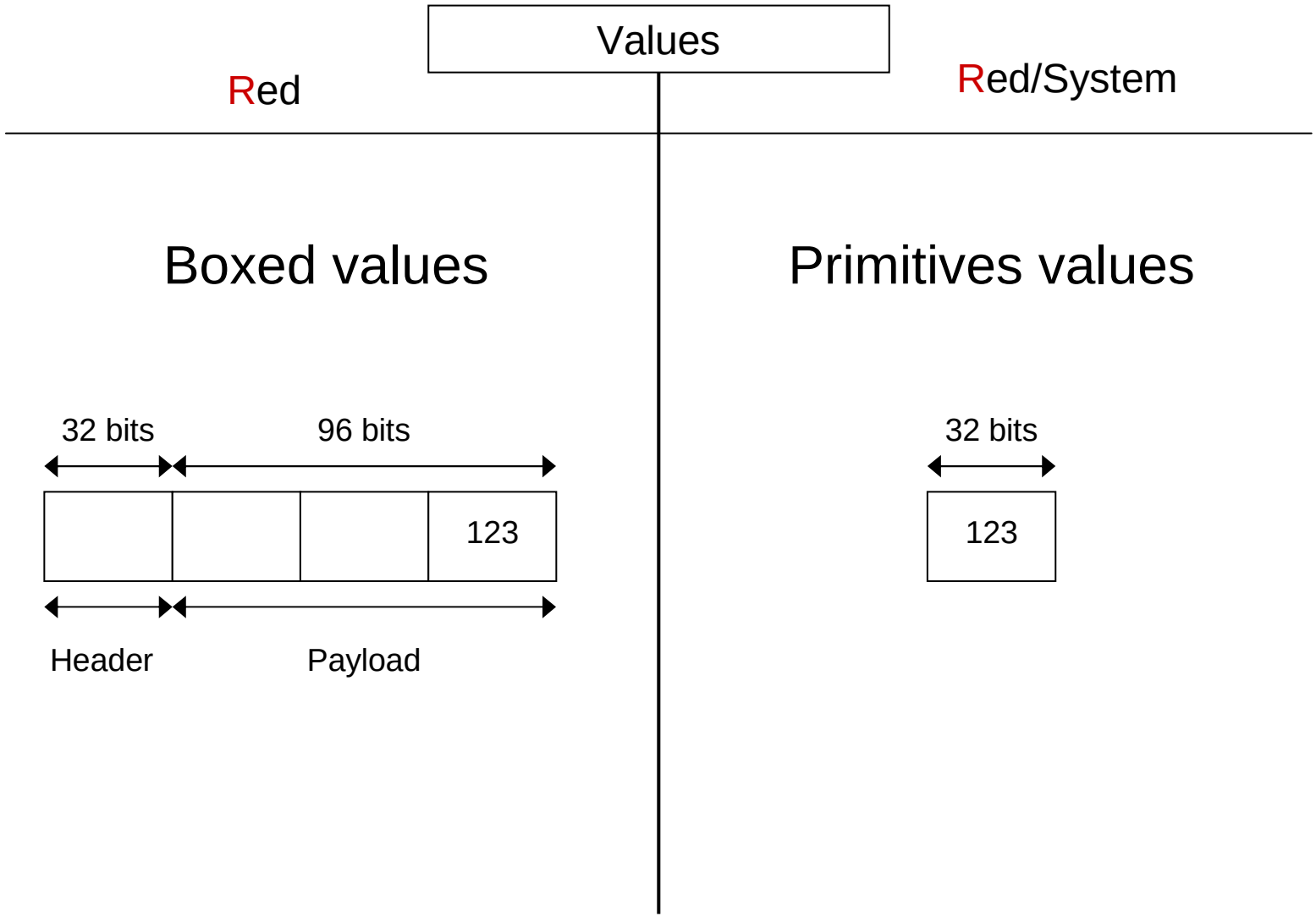
Red quick tour: Red vs Red/System (2/6)



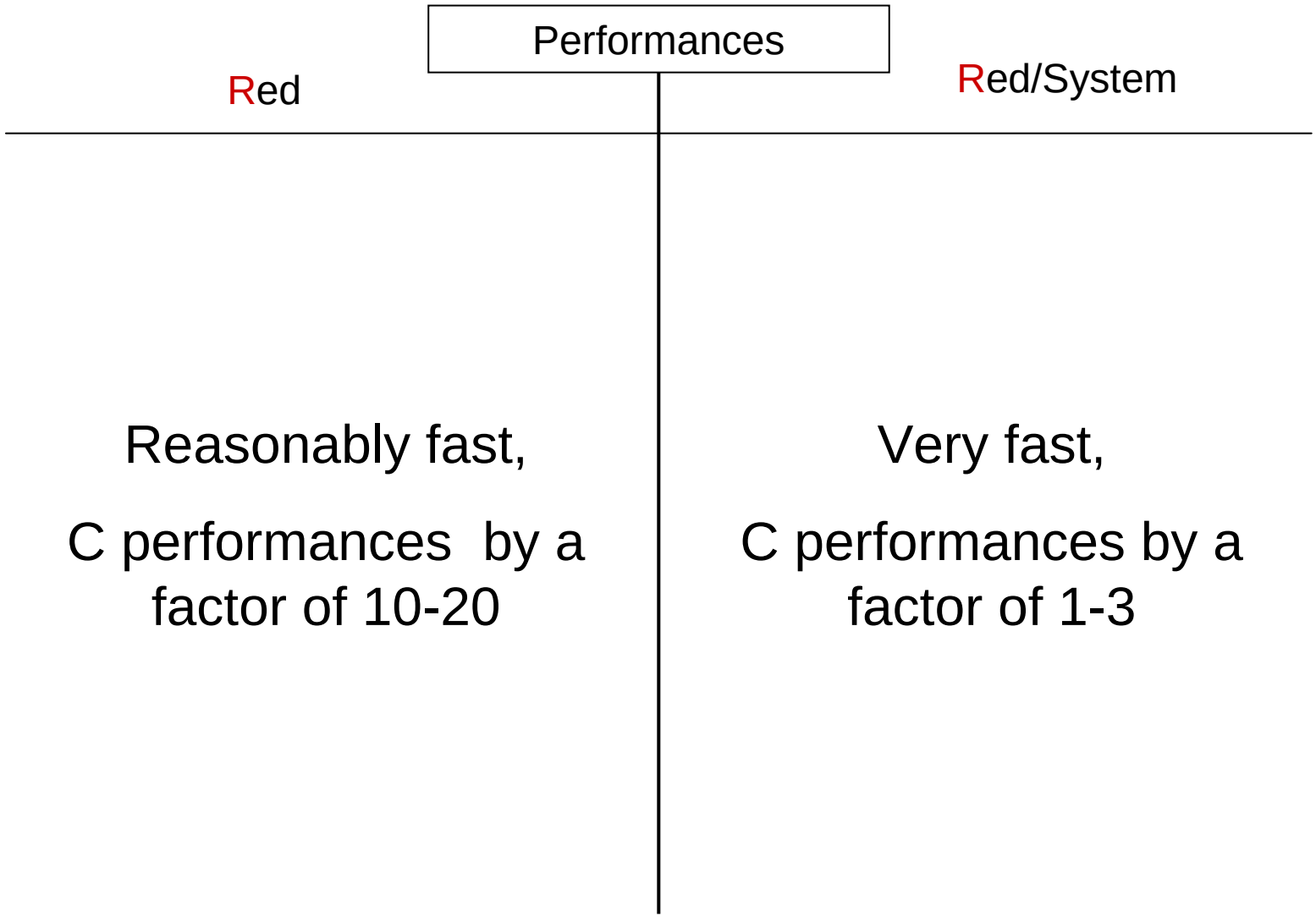
Red quick tour: Red vs Red/System (3/6)



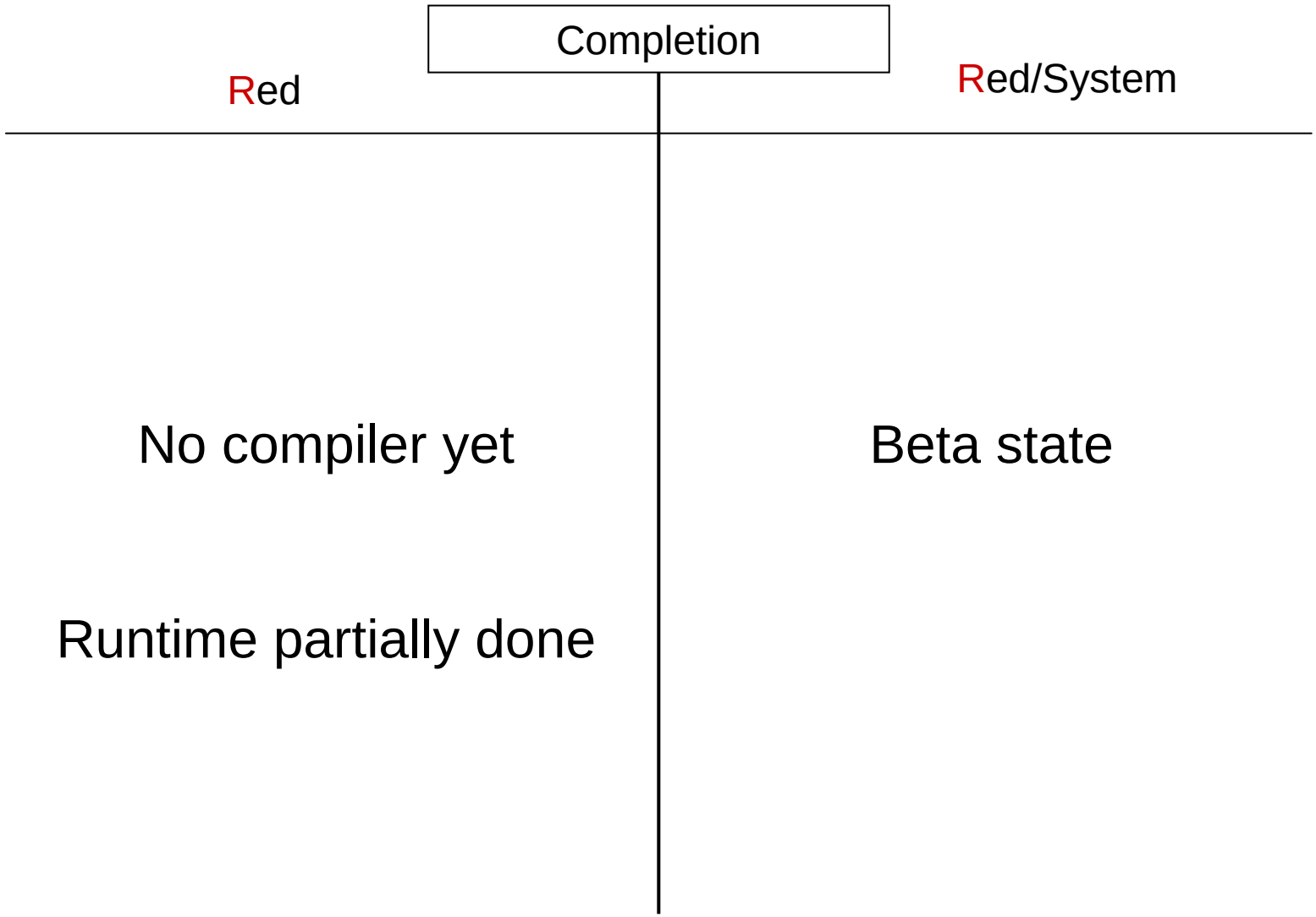
Red quick tour: Red vs Red/System (4/6)



Red quick tour: Red vs Red/System (5/6)



Red quick tour: Red vs Red/System (6/6)



Red quick tour: Goals (1/7)

Simplicity

An IDE should not be necessary to write code.

Red quick tour: Goals (2/7)

Compactness

Being highly expressive maximizes productivity.

Red quick tour: Goals (3/7)

Speed

If too slow, it cannot be general-purpose enough.

Red quick tour: Goals (4/7)

Be Green

Have a Small Footprint

Because resources are not limitless.

Red quick tour: Goals (5/7)

Ubiquity

Spread everywhere.

Red quick tour: Goals (6/7)

Portability

Write once run everywhere

That’s the least expected from a programming language.

Red quick tour: Goals (7/7)

Flexibility

~~*Best*~~ *Good fit for any task!*

Red quick tour: Some features…

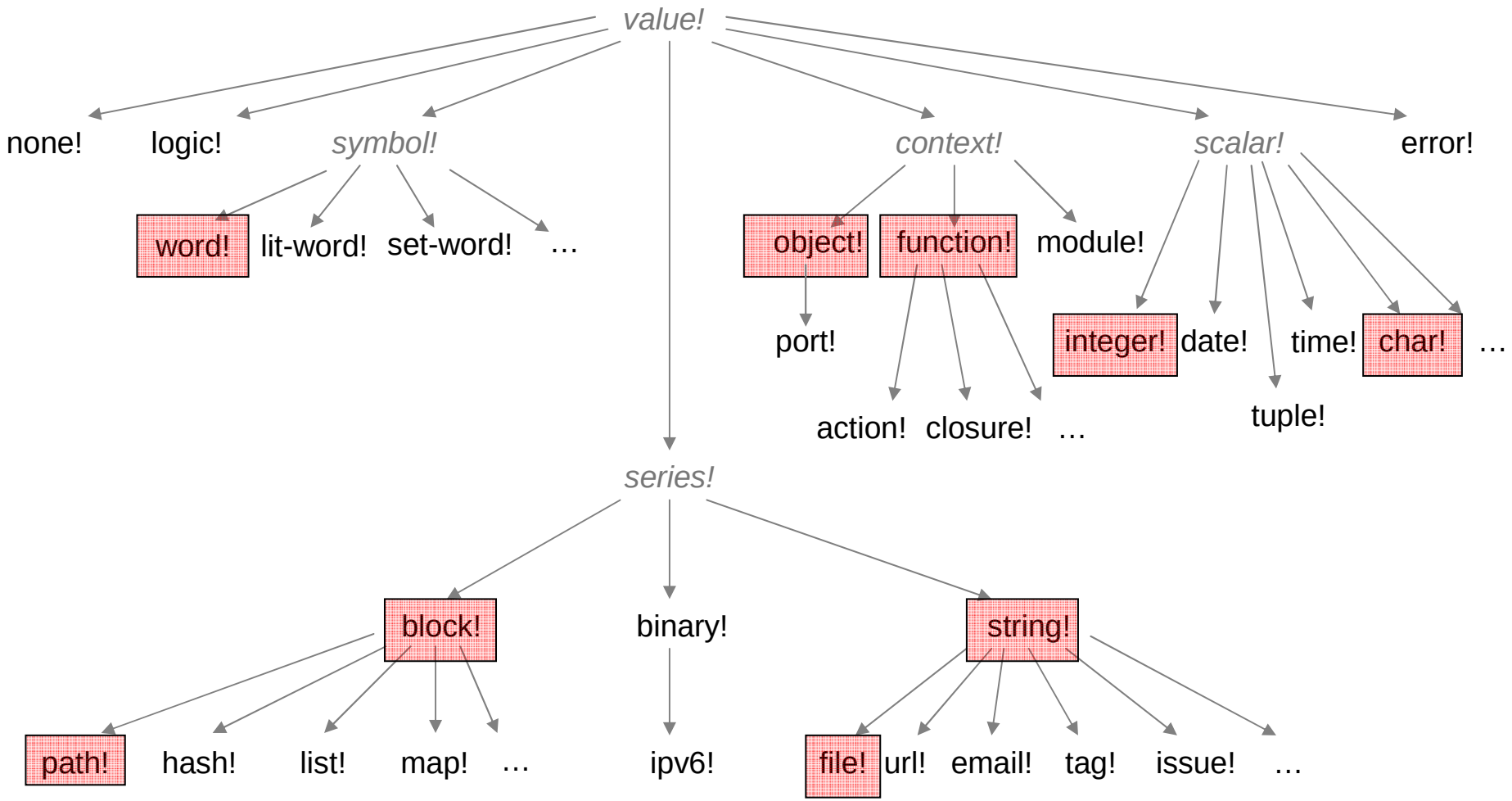
- Same syntax as REBOL, most of its semantic rules
- Strong DSL / dialecting support
- Red/System dialect inlined in Red

- (JIT) compiled instead of interpreted
- Statically typed + type inference
- Embeddable: distributed as a shared library with host languages bindings

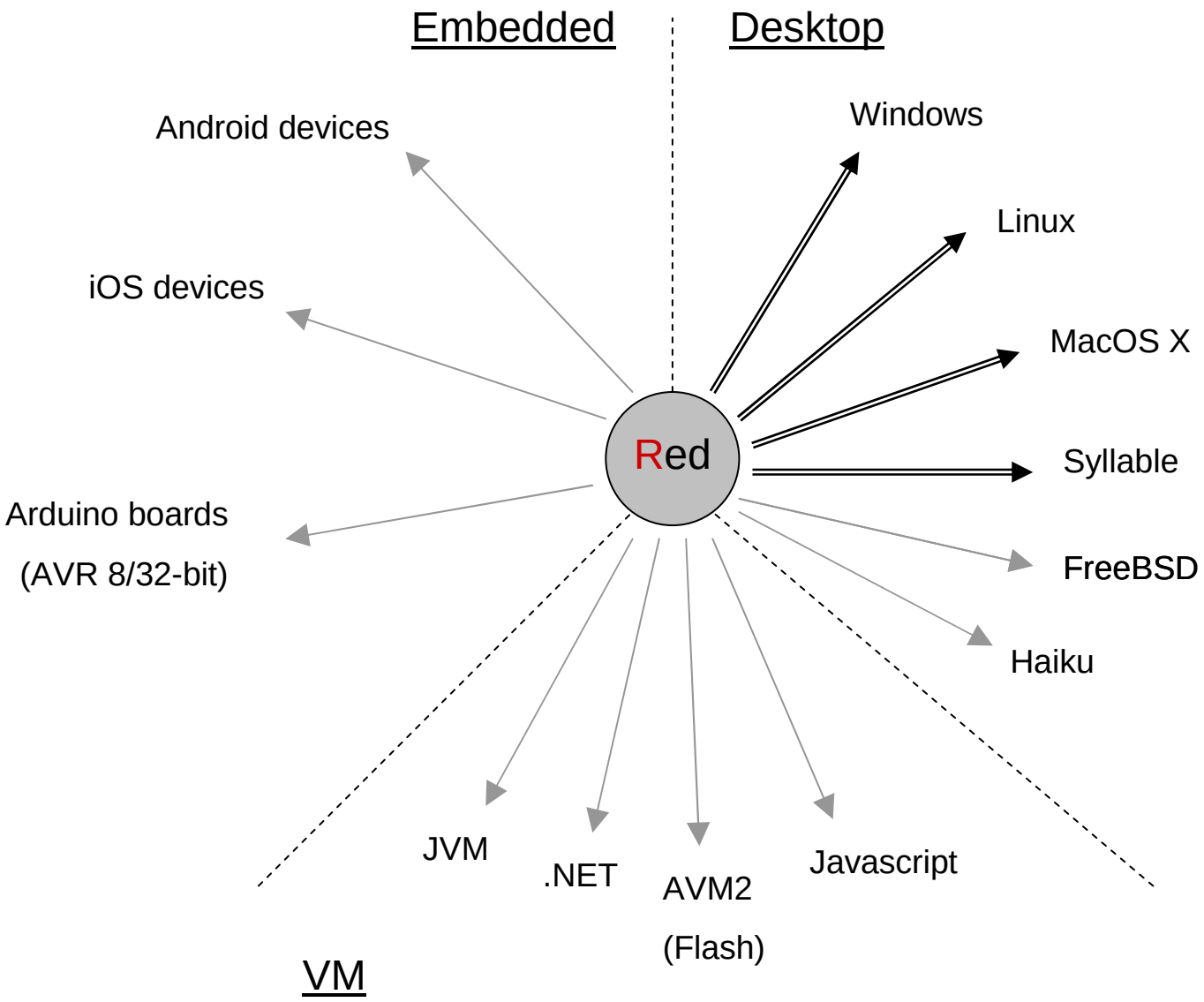
- Concurrency support
  - Task parallelism: using "actor" abstraction
  - Data parallelism: using parallel collections

Red quick tour: Types tree

Rich type system: up to 50 first-class datatypes



Red quick tour: Target platforms



Red quick tour: bootstrapping

1. Red and Red/System compilers written in REBOL
2. Red/System compiler rewritten in Red
3. Red compiler rewritten in Red
4. Red JIT-compiler written in Red

Red quick tour: Project

- BSD license

- Source code hosted on Github since March 2011
  - version 0.2.1, 3 commiters, 537 public commits
  - 175 tickets (164 closed)
  - 8614 unit tests (framework built by Peter WA Wood)
  - 260 KiB of sources for Red/System
  - 3800 LOC for Red/System compiler
  - 2200 LOC for Red/System linker

Red quick tour: Planning

- ~~Sept.~~ October 2011:
  - ~~beta~~ alpha of Red (no JIT)
  - ~~alpha~~ beta of ARM support
  - alpha of the IDE

- ~~Dec.~~ January 2012:
  - ~~V1.0~~ beta of Red (no JIT)
  - beta of the IDE

- Q1 2012:
  - beta of Red JIT-compiler
  - V1.0 of Red
  - v1.0 of IDE

Red online channels

- Home: http://red-lang.org
- Twitter: #red_lang
- IRC channel: #red-lang on freenode
- Mailing list hosted on Google Groups

Red/System

Red/System: features (1/2)

- Purely imperative, C-level language, with a Red syntax
- Statically compiled (naïve compilation for now)
- Limited type system:
  - Logic!, byte!, integer!, struct!, pointer!, c-string!
  - Simple type inference
  - Type casting supported
- Compiler directives: `#define, #include, #import, #syscall, #if, #either, #switch,…`
- Low-level CPU support (interruptions, I/O, stack, privileged mode)
- *Inlined ASM support*

Red/System: features (2/2)

- Linker
  - Link-time shared libraries binding
  - Output types: Exe, Shared library, Static library
  - Formats: PE, ELF, mach-o, Intel-hex
  - Link third-party static libraries
- Targets: IA-32, ARM, JVM, AVM2, x64, CLR
- Red/System as an inlined dialect in Red

Red/System: Hello world!

```
Red/System [
    title: "Hello World demo"
]

print "hello world!"
```

Red/System: variables and expressions

```
a: 1
b: a + 2 * 4
c: a < b
d: "hello"

if a < b [print "b is greater"]
either a < b [print "b"][print "a"]

print either a < b ["b"]["a"]

print [a " is less than " b "," c "," d]
1 is less than 12,true,hello
print-wide [a "is less than" b c d]
1 is less than 12 true hello
```

Red/System: functions

```
nothing: function [][]

print-zero: function [n [integer!]][
    print either zero? n ["zero"]["not zero"]
]

abs: function [n [integer!] return: [integer!]][
    either positive? n [n][negate n]
]

uppercase: function [s [c-string!] /local offset][
    offset: #"a" - #"A"
    if any [#"a" <= b/1   b/1 <= #"z"][
            s/1: s/1 + offset
    ]
]
```

Red/System: shared libraries

```
#import [
    "libc.so.6" cdecl [
            allocate: "malloc" [
                    size    [integer!]
                    return: [byte-ptr!]
            ]
            free: "free" [
                    block   [byte-ptr!]
            ]
            quit:  "exit" [
                    status  [integer!]
            ]
            printf: "printf" [[variadic]]
    ]
]
printf ["%i %s" 123 "hello"]
123 hello
```

Red/System: CPU low-level features

```
timer-handler: func [[interrupt]][...]

#interruptions [
     0000h   :reset
     0004h   :timer-handler
]

a: read-io 0376h
write-io 0376h 1

a: get-modes privileged
set-modes privileged false

set-modes interrupt true
set-modes interrupt-mask FF000000h
```

Red/System: keywords

%	*	+
-	-**	/
//	///	<
<<	<=	<>
=	>	>>
>=	>>>	alias
all	and	any
as	assert	comment
declare	either	exit
false	func	function
if	not	or
pop	push	return
size?	true	until
while	xor	
switch	case	repeat
loop	set-modes	get-modes
read-io	write-io	

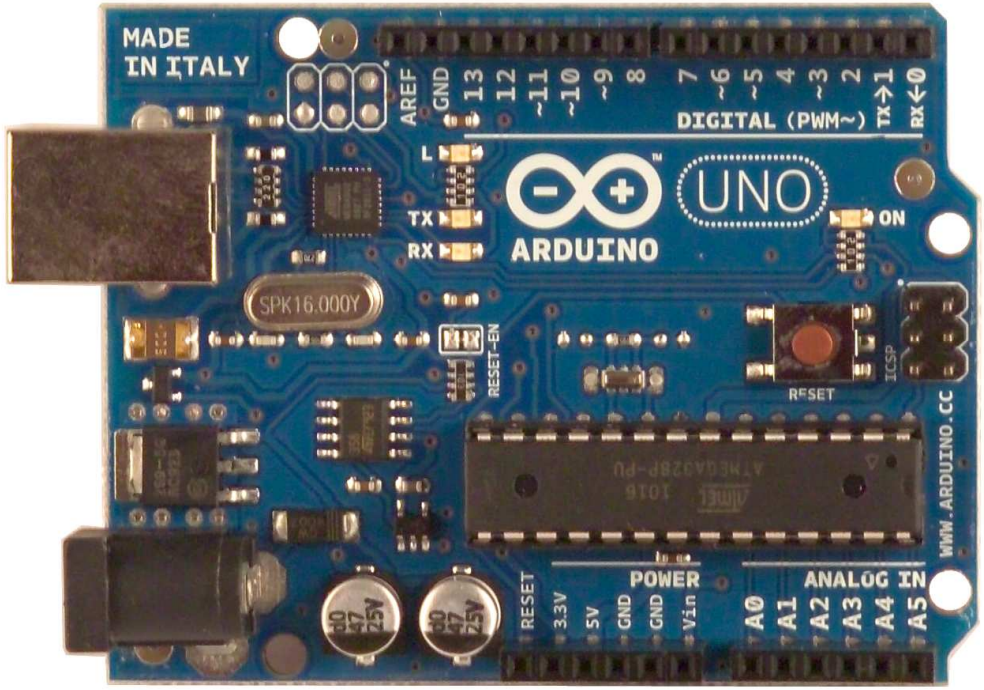
Red/System: library bindings

- C library binding
- cURL binding
- ZeroMQ binding
- SDL binding
- GTK binding

All written by Kaj de Vos.

Let see a few demos written with these bindings…

Arduino Uno



Microcontroller	ATmega328
Flash Memory	**32 KB** of which 0.5 KB used by bootloader
SRAM	**2 KB**
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz