Software Component Synthesis Tools

Dick Hamlet

Department of Computer Science

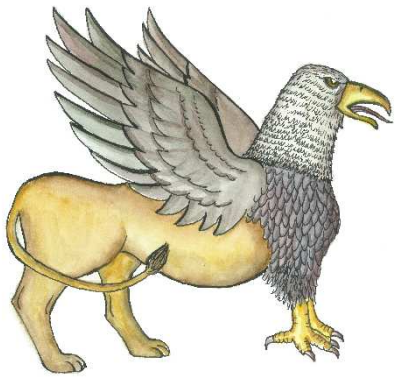
Portland, OR 97207 USA

hamlet@cs.pdx.edu

Supported by NSF CCR-0112654 and SFI E.T.S. Walton Fellowship

From a Modern Beastiary

THE GRIFFIN is a fabulous animal with the body of a lion and the head of an eagle. The griffin is powerful and fierce because of its parts.



It will tear to pieces any human being it comes across. Software developers should pay attention to this, because software also may tear people to pieces. Software made from components has the properties of its parts.

NSFAQ

- What are software tools? (Who builds them?)
- What is a software component? (And why bother?)
- How is software tested?
- How are components combined into systems? (Do the systems work?)

Software Development "Tools"

TOOL: An instrument for performing mechanical operations; anything used like a tool to effect some result.

Hence, programs that are used to help develop other programs. For example, an editor, a compiler, a linker, etc.

This talk is about Perl programs (mostly written by high school interns!) that help develop component-based software systems.

Software Components: What and Why

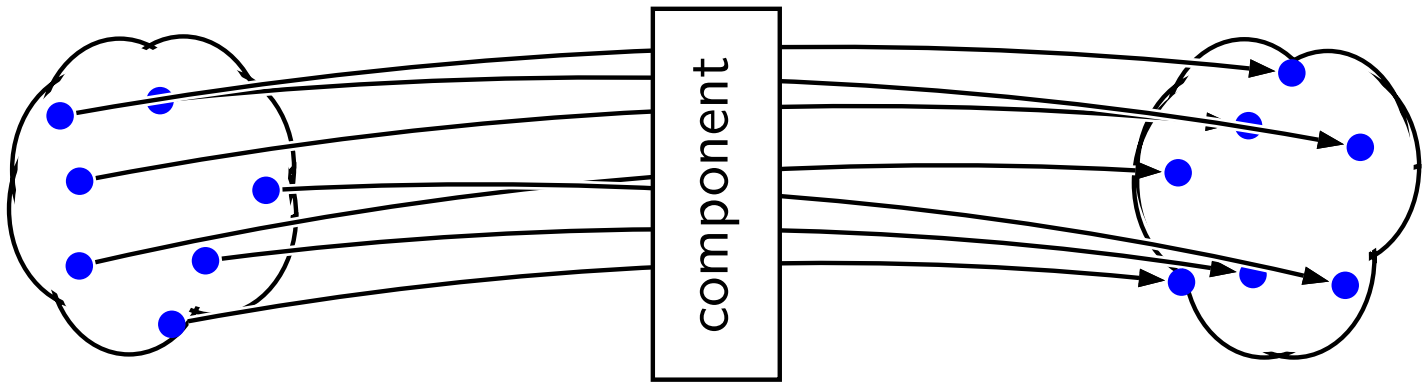
Definition: A *software component* is an executable program that obeys a set of conventions for its interface with other programs.

Rationale:

- Components can be developed (*and tested*) in isolation, then later assembled into systems.
- System code is easier to understand because its "statements" are its components.
- If the components work properly, the system is more likely to work, too.

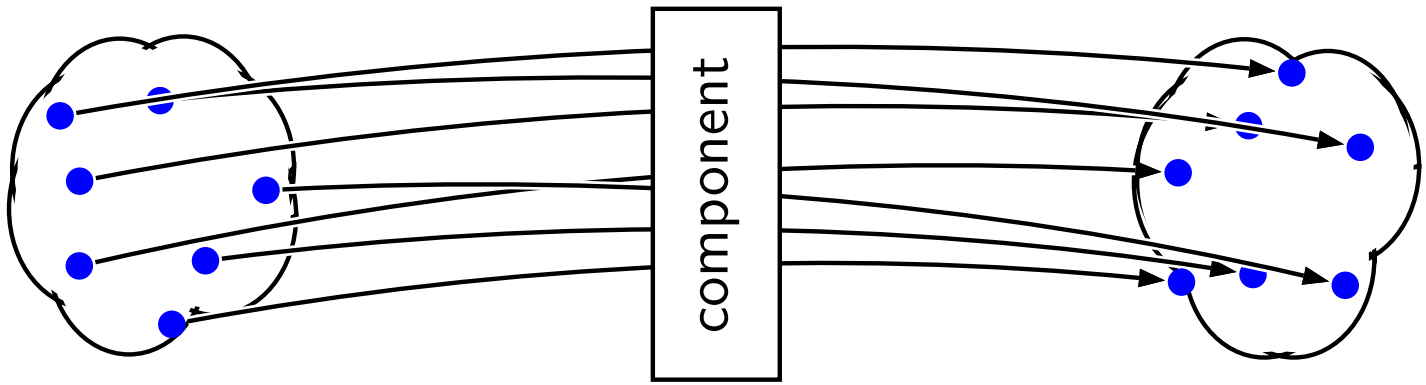
Subdomain Testing

Testing a component:

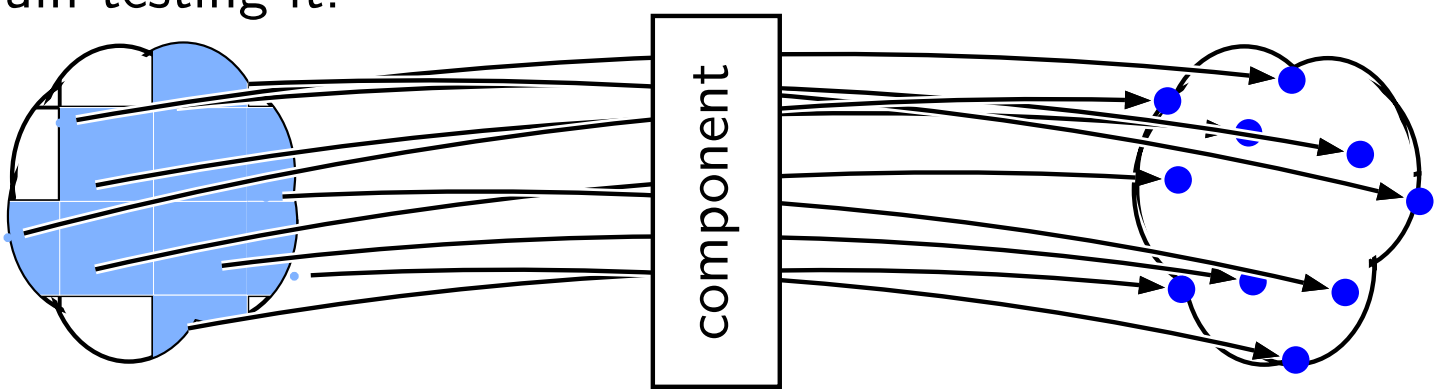


Subdomain Testing

Testing a component:



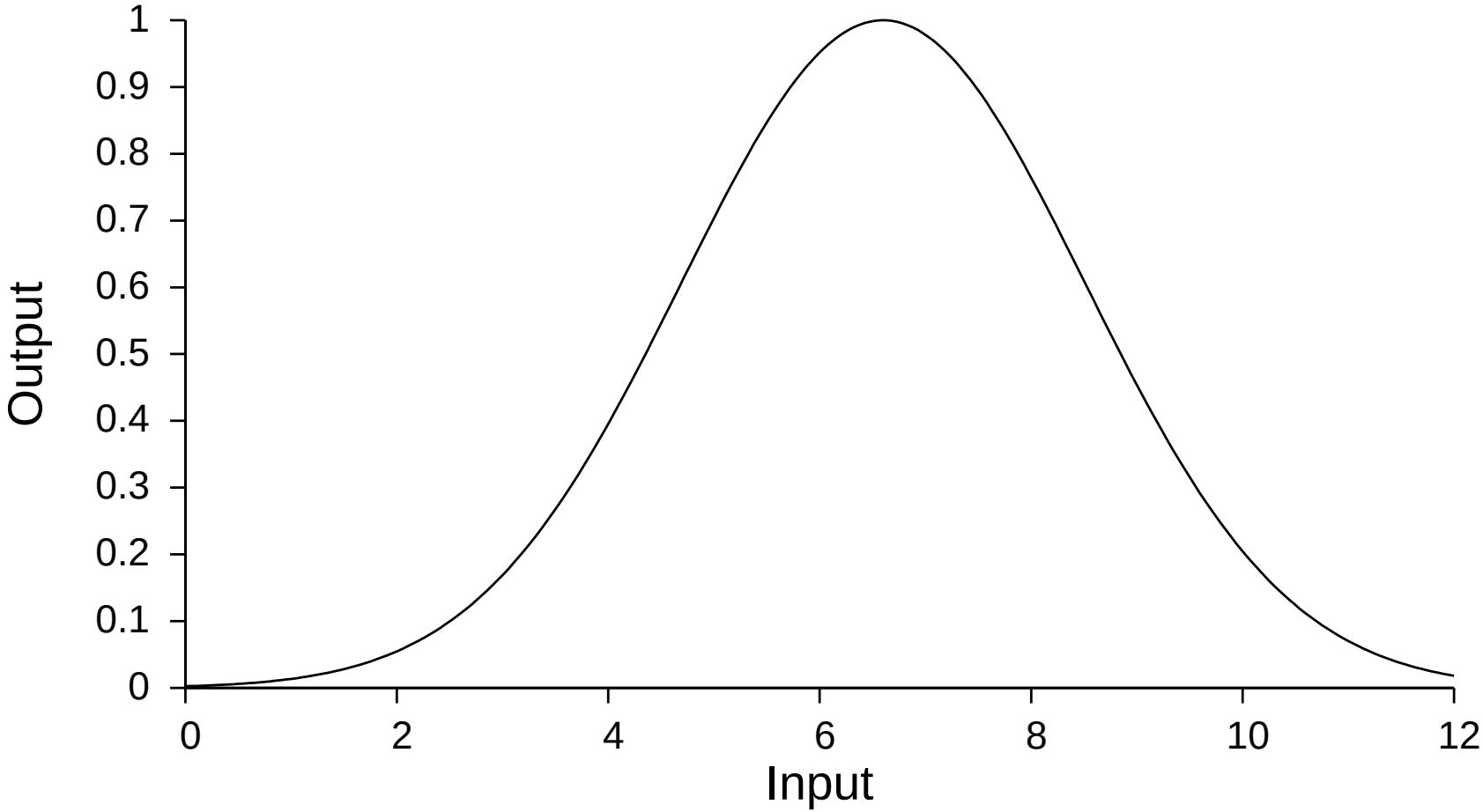
Subdomain testing it:



Testing a Simple Component: `Bell`

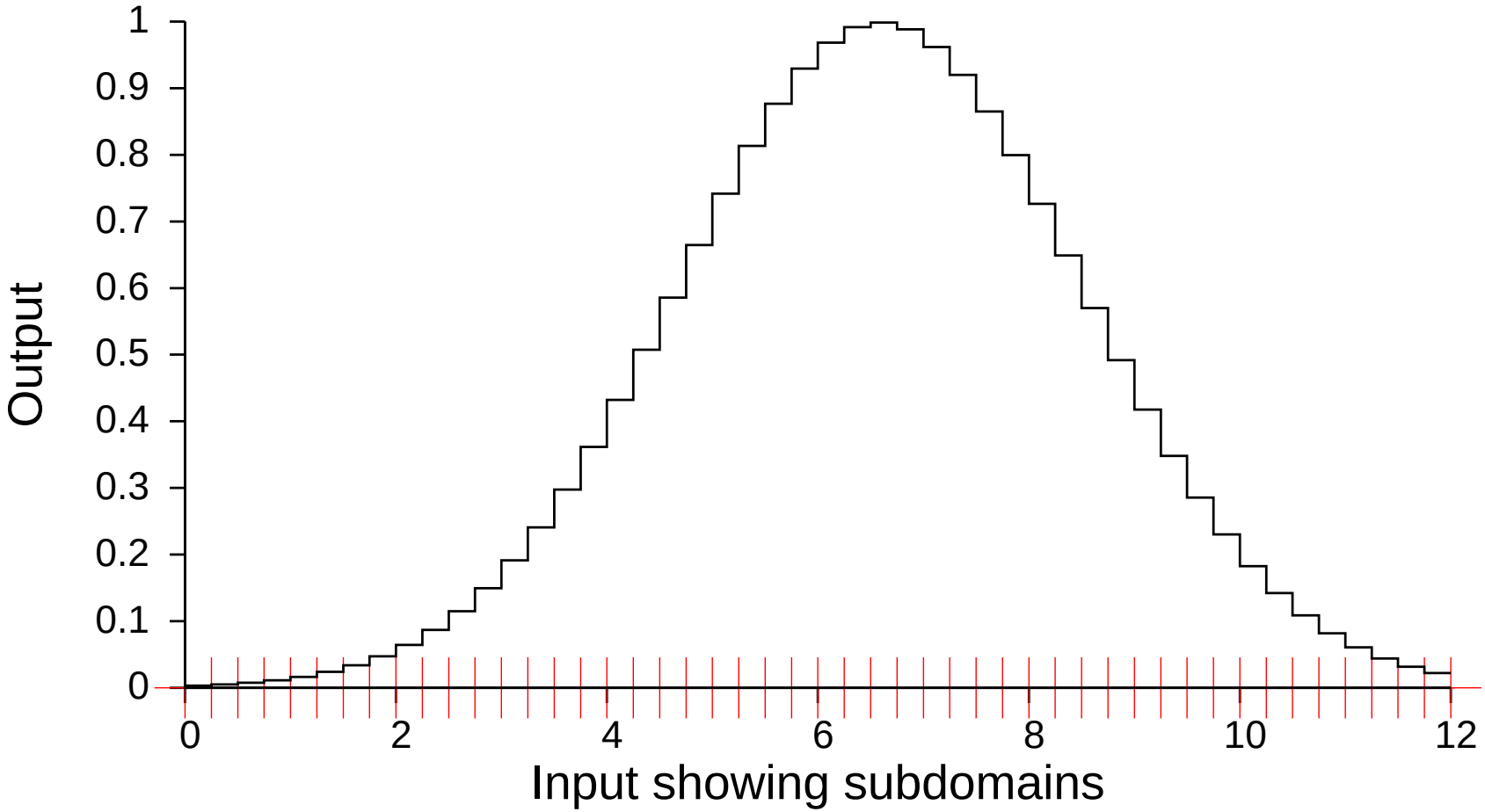
Component '`Bell`' measurement

Gaussian, $\mu = 6.6, \sigma = 2.7$, 142 equispaced tests:



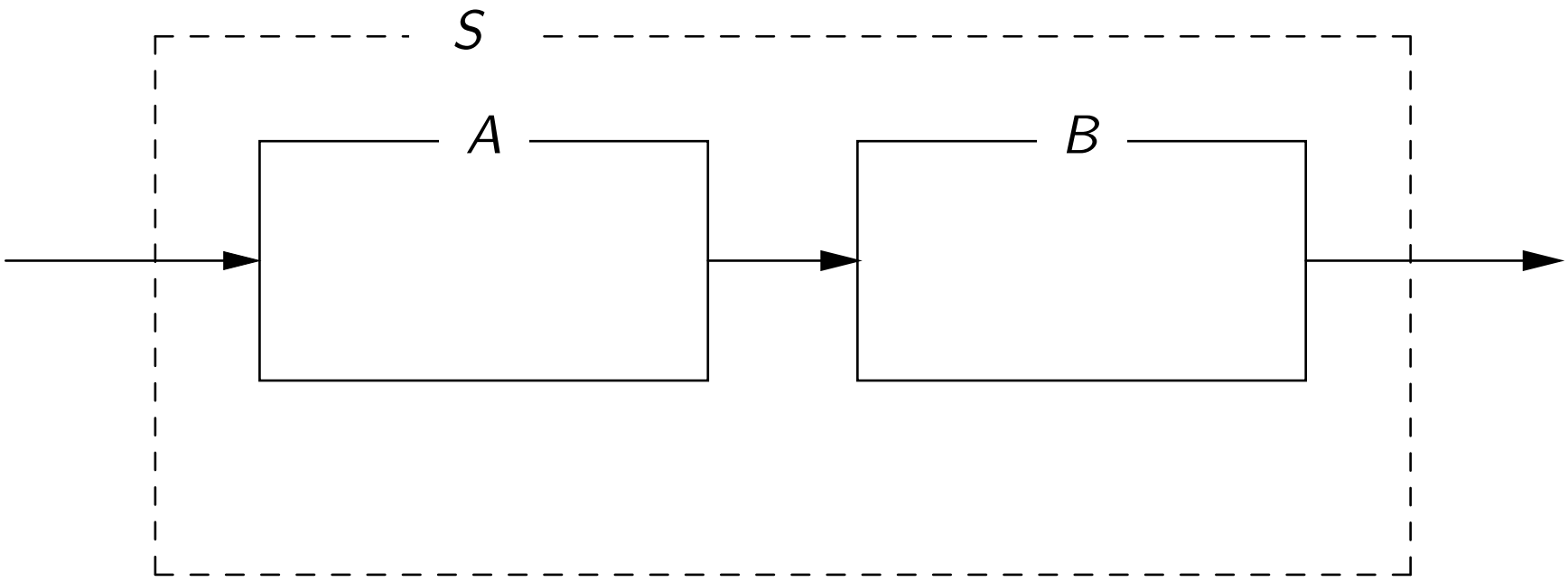
Testing a Simple Component: `Bell`

Approximation measurements
(Average of 3 samples in each of 48 subdomains):

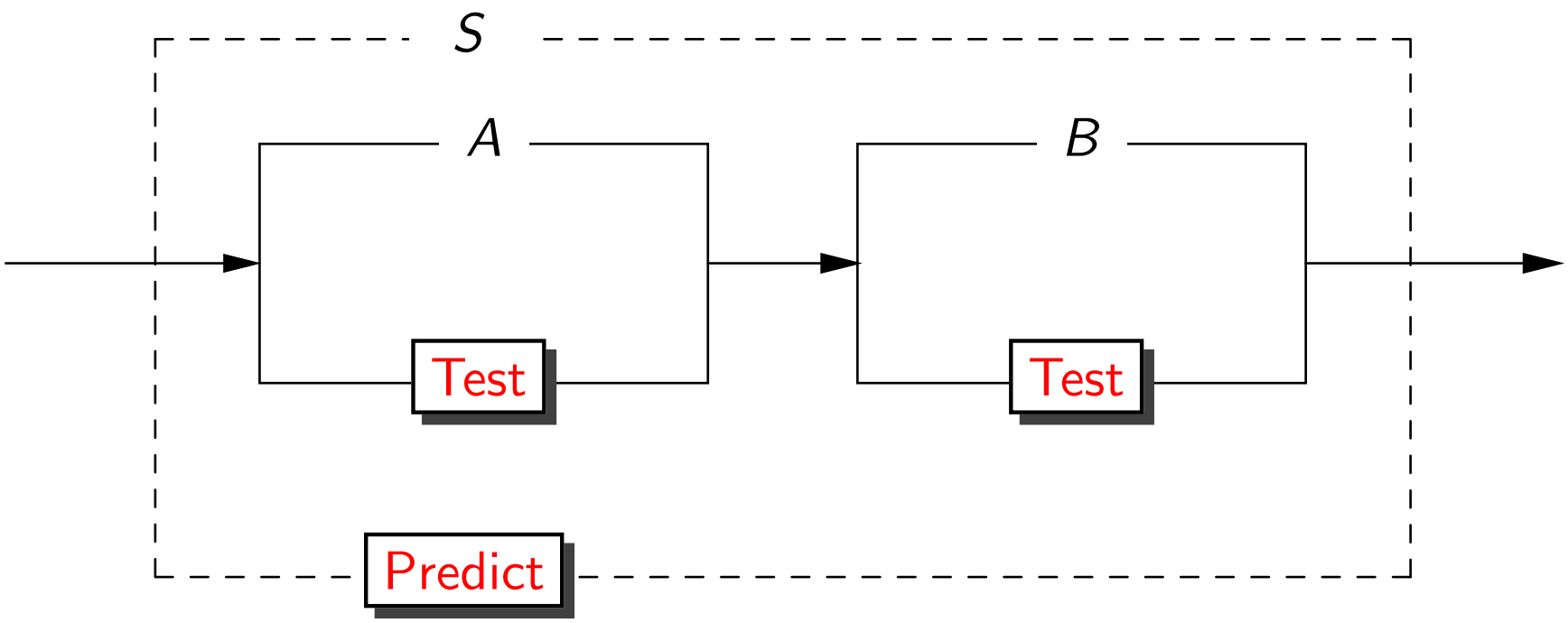


R-M-S error ranges from 0.1% to 4.3% by subdomain

Composing Components

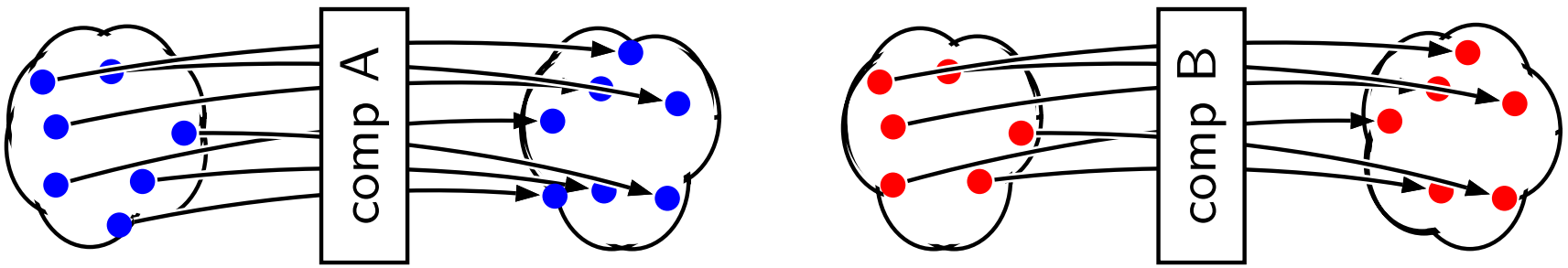


Composing Components



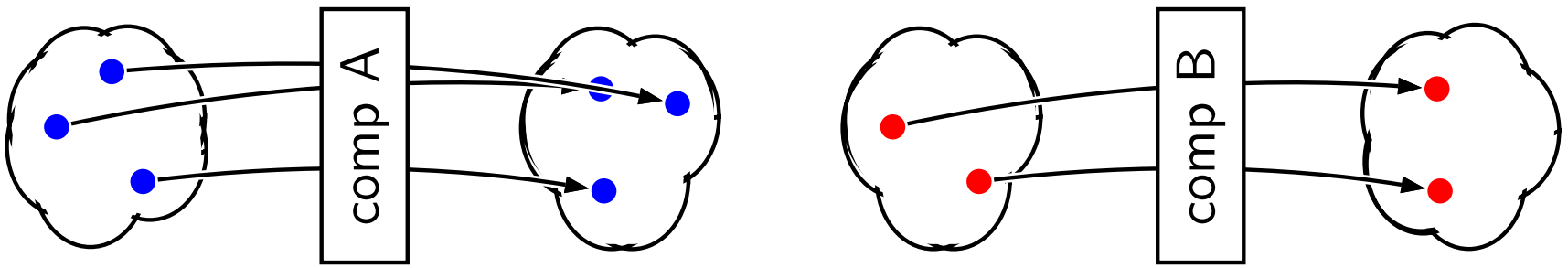
Tests Don't Compose

Testing two components A and B separately:



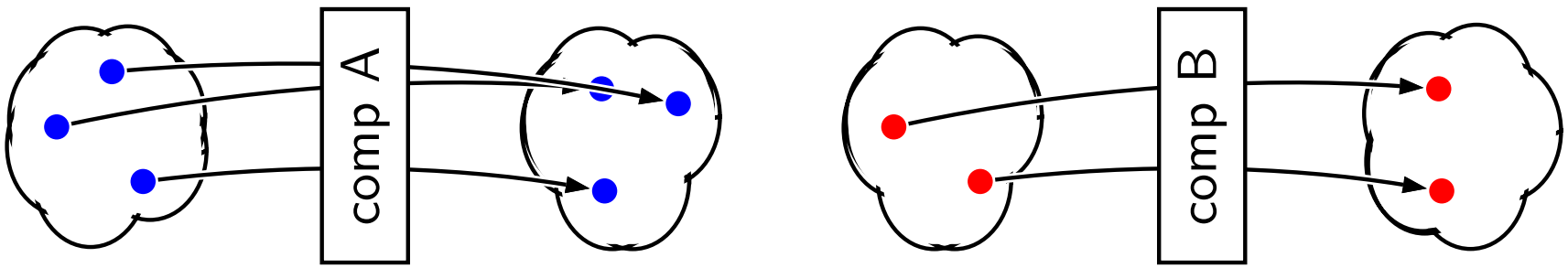
Tests Don't Compose

Testing two components A and B separately:

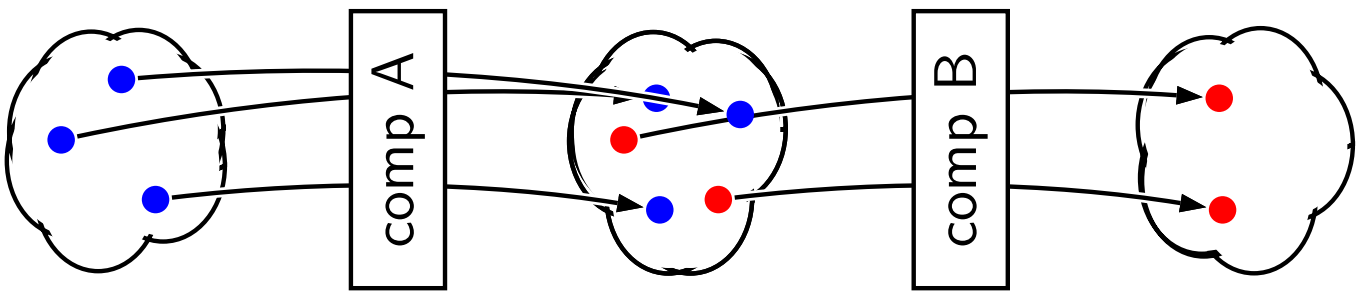


Tests Don't Compose

Testing two components A and B separately:

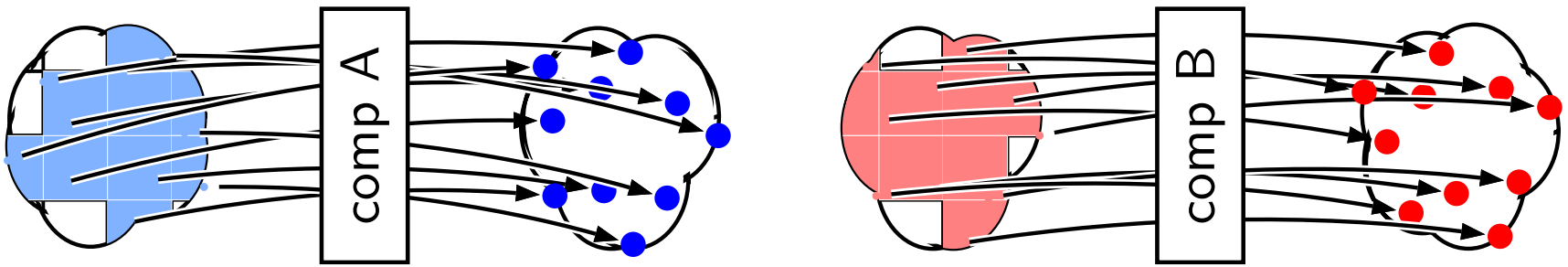


Series combination doesn't match at the interface



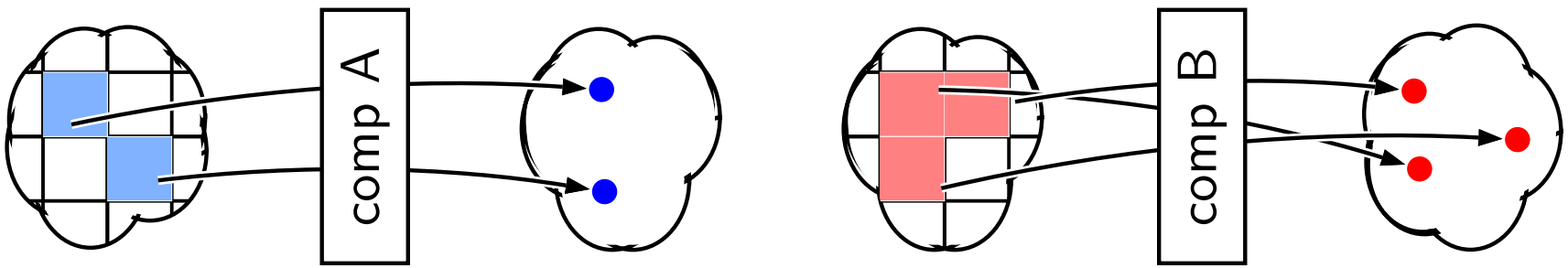
Subdomains to the Rescue!

Subdomain testing two components A and B separately:



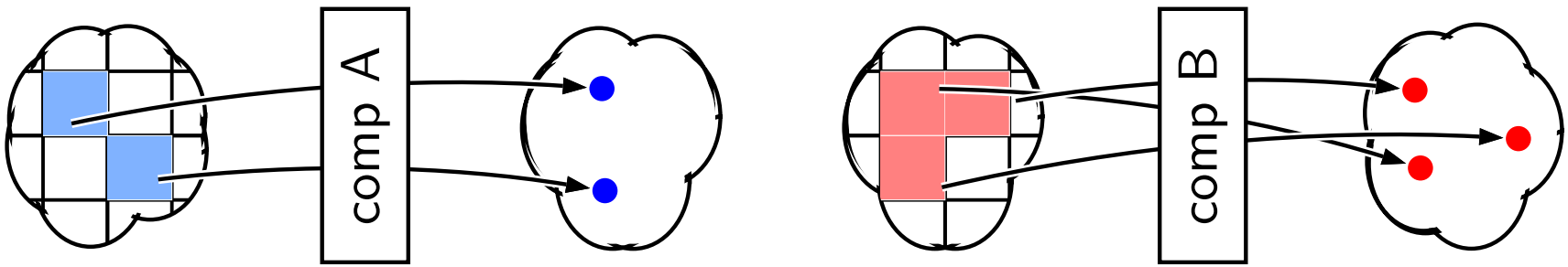
Subdomains to the Rescue!

Subdomain testing two components A and B separately:

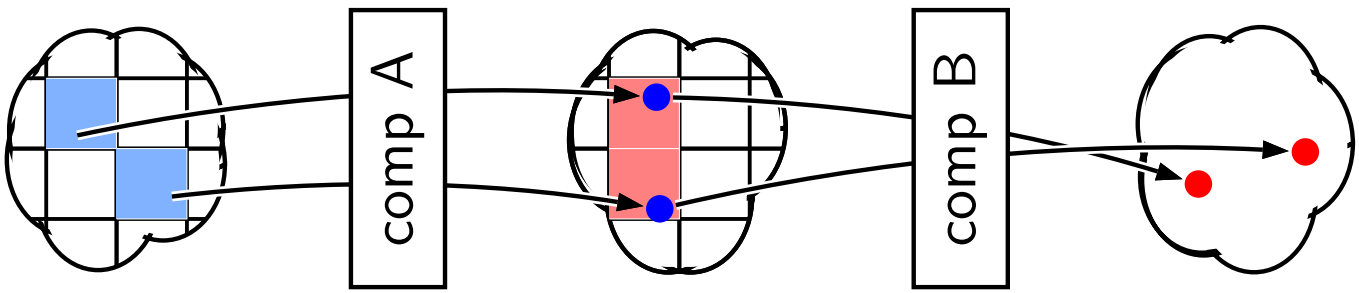


Subdomains to the Rescue!

Subdomain testing two components A and B separately:

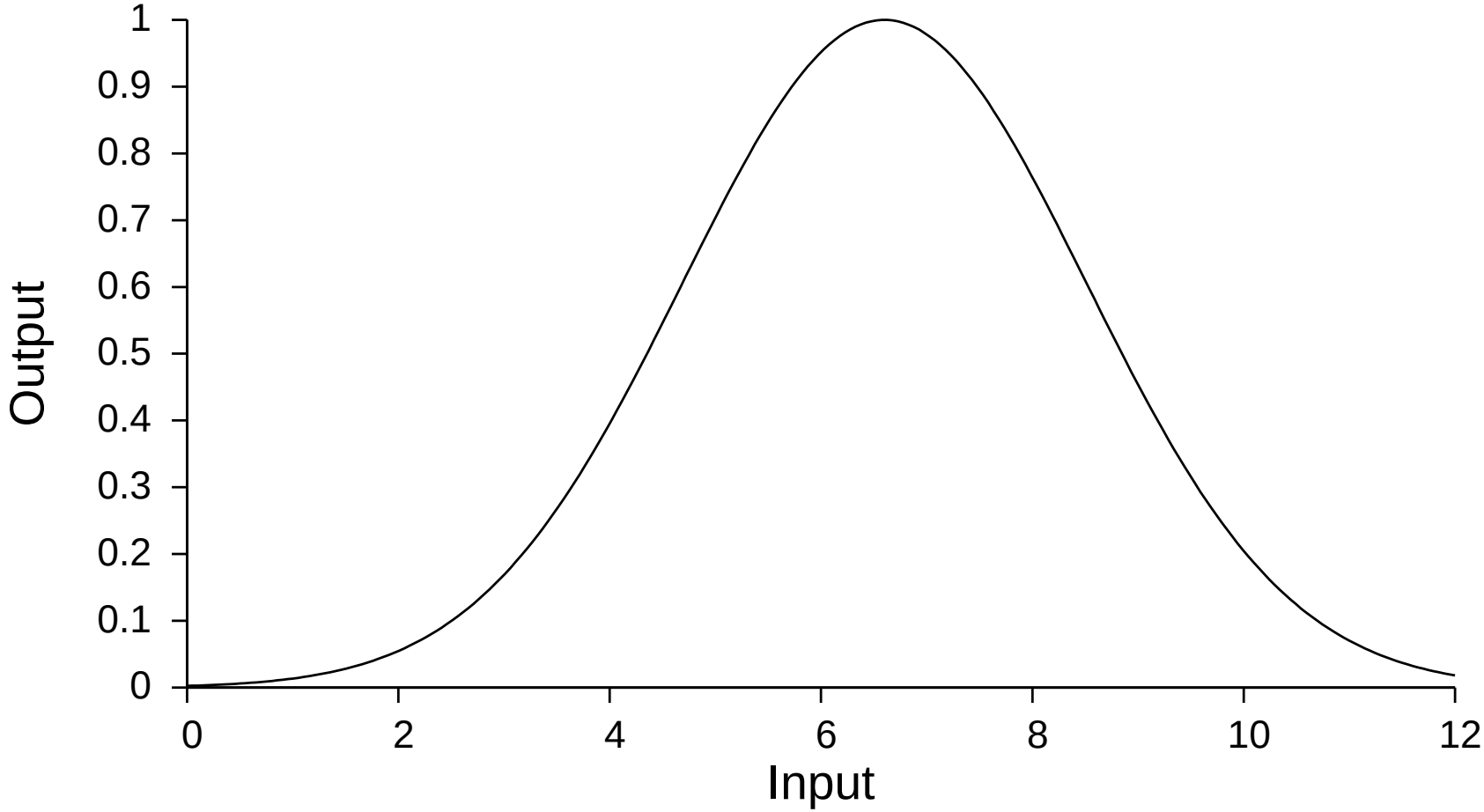


Never an interface mismatch!

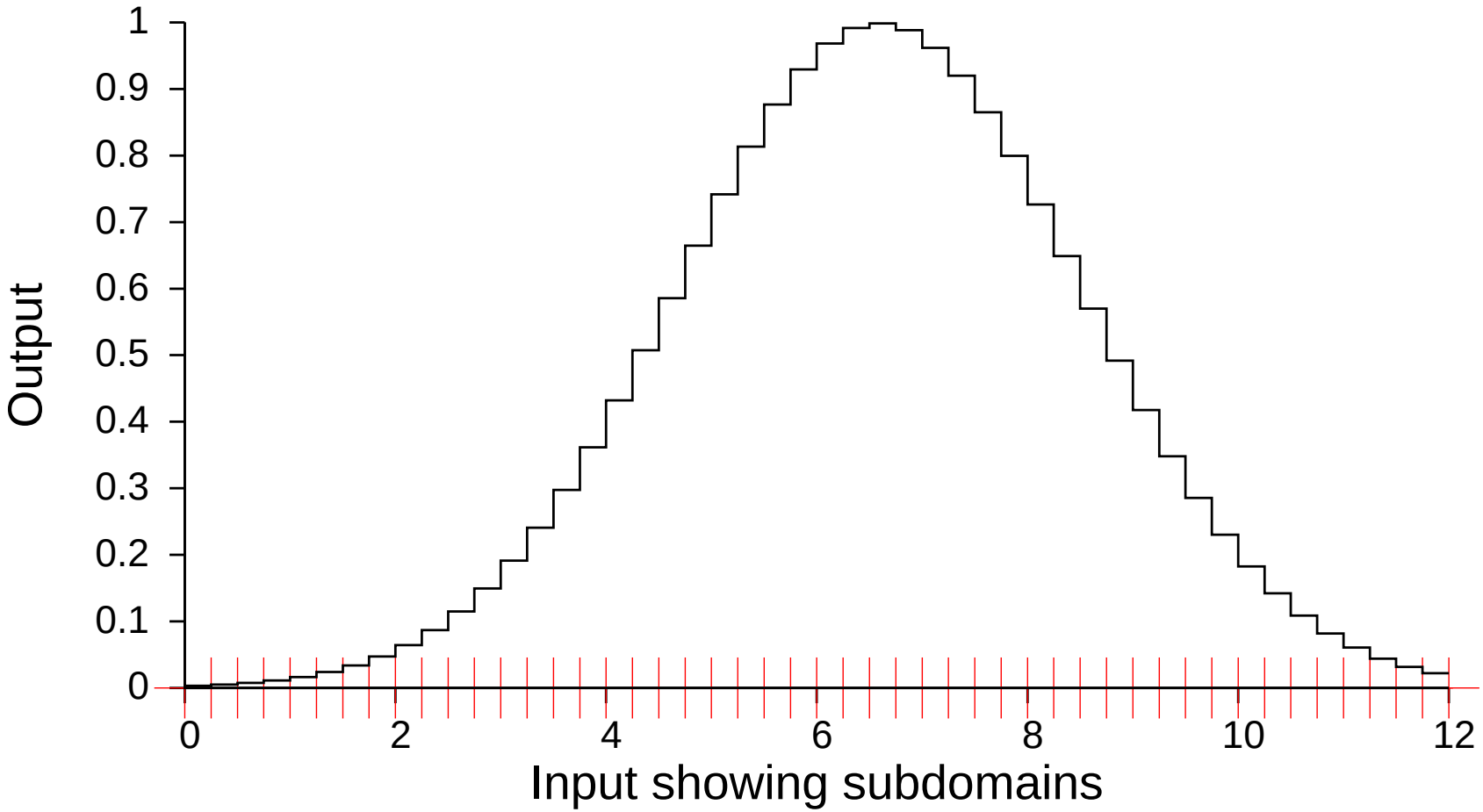


Example: `Chop; Bell`

You remember '`Bell`':



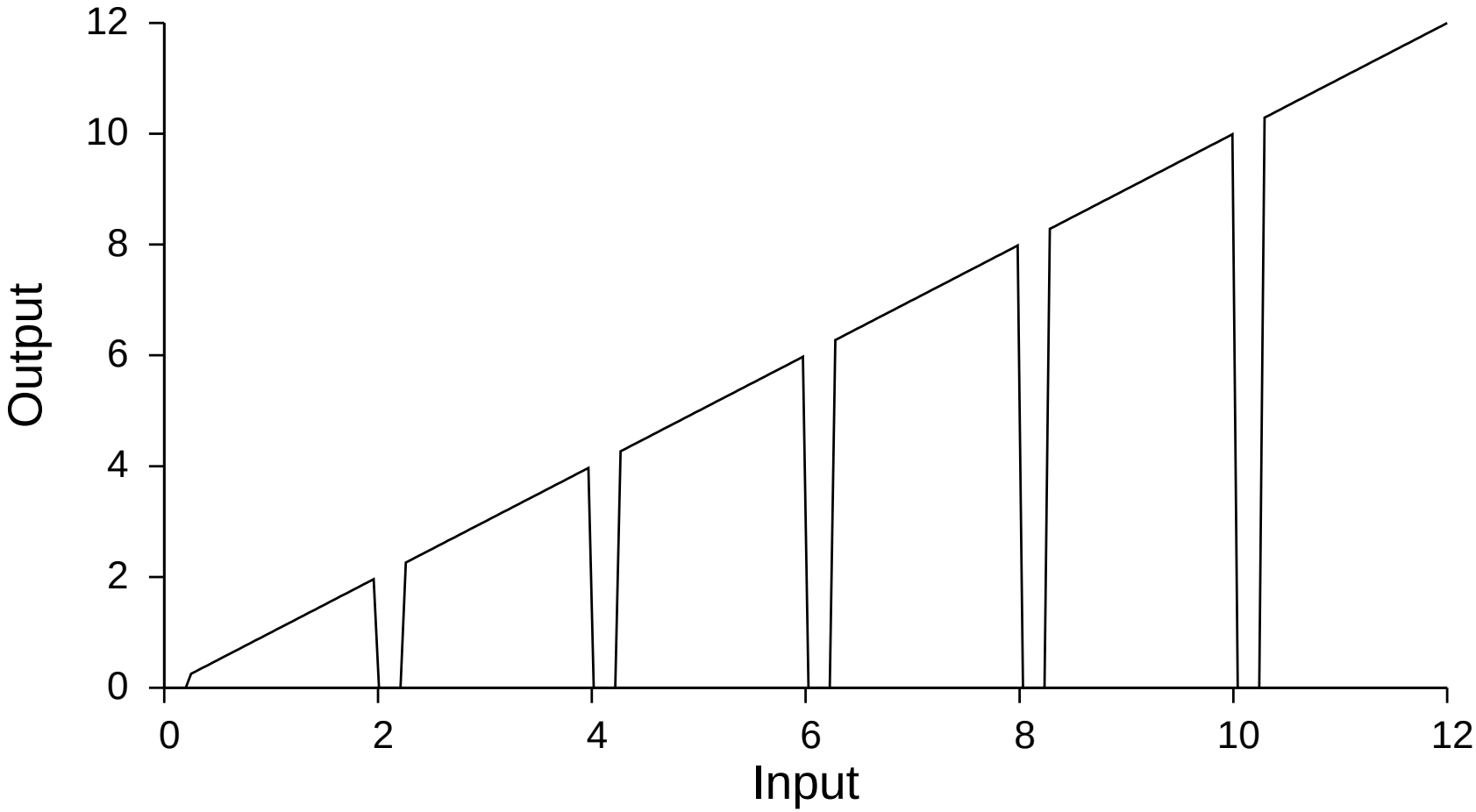
Example: `Chop; Bell`



Example: `Chop; Bell`

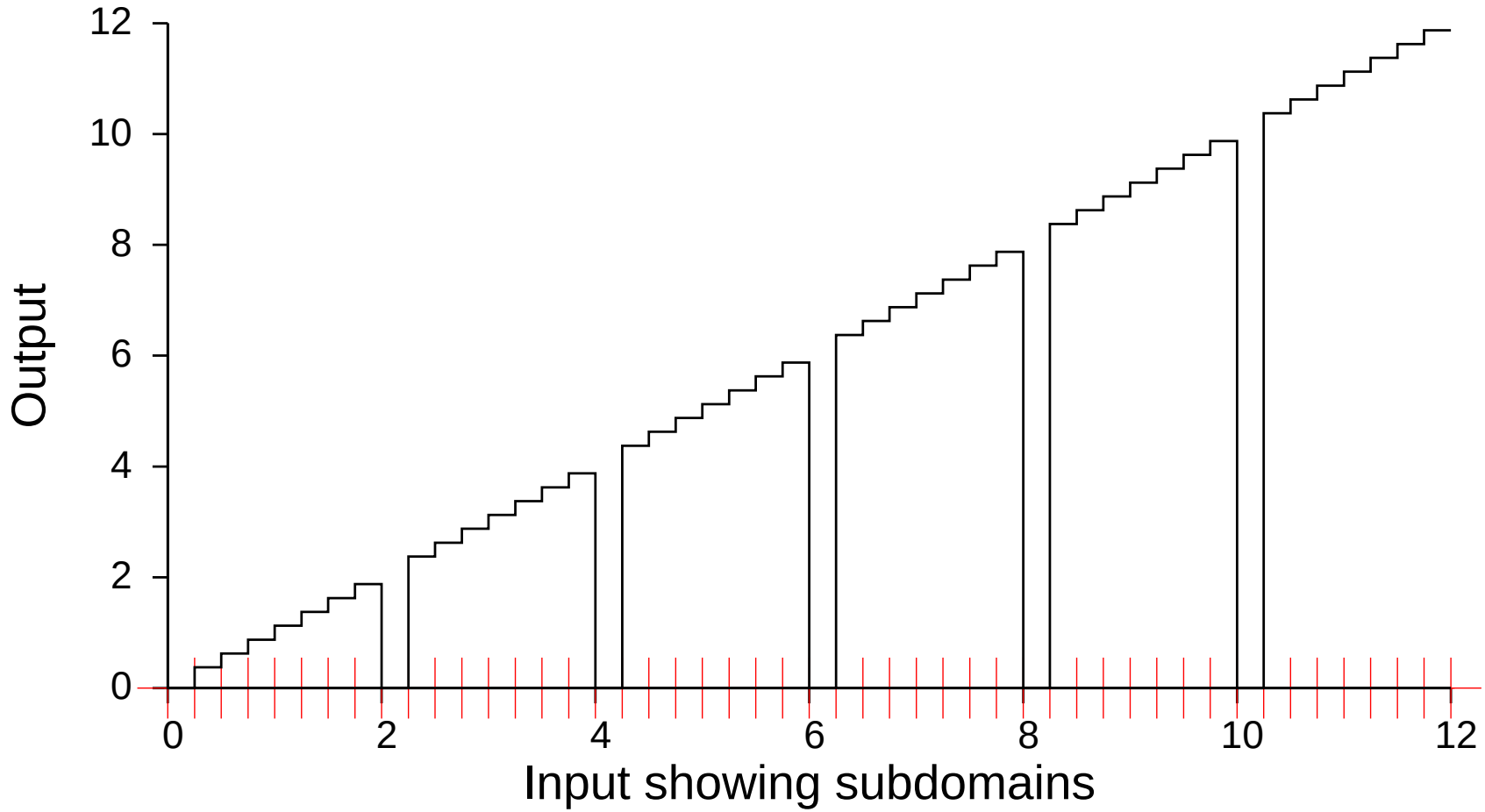
Component '`Chop`' measurement:

Chopping function, 142 equispaced tests:



Example: `Chop; Bell`

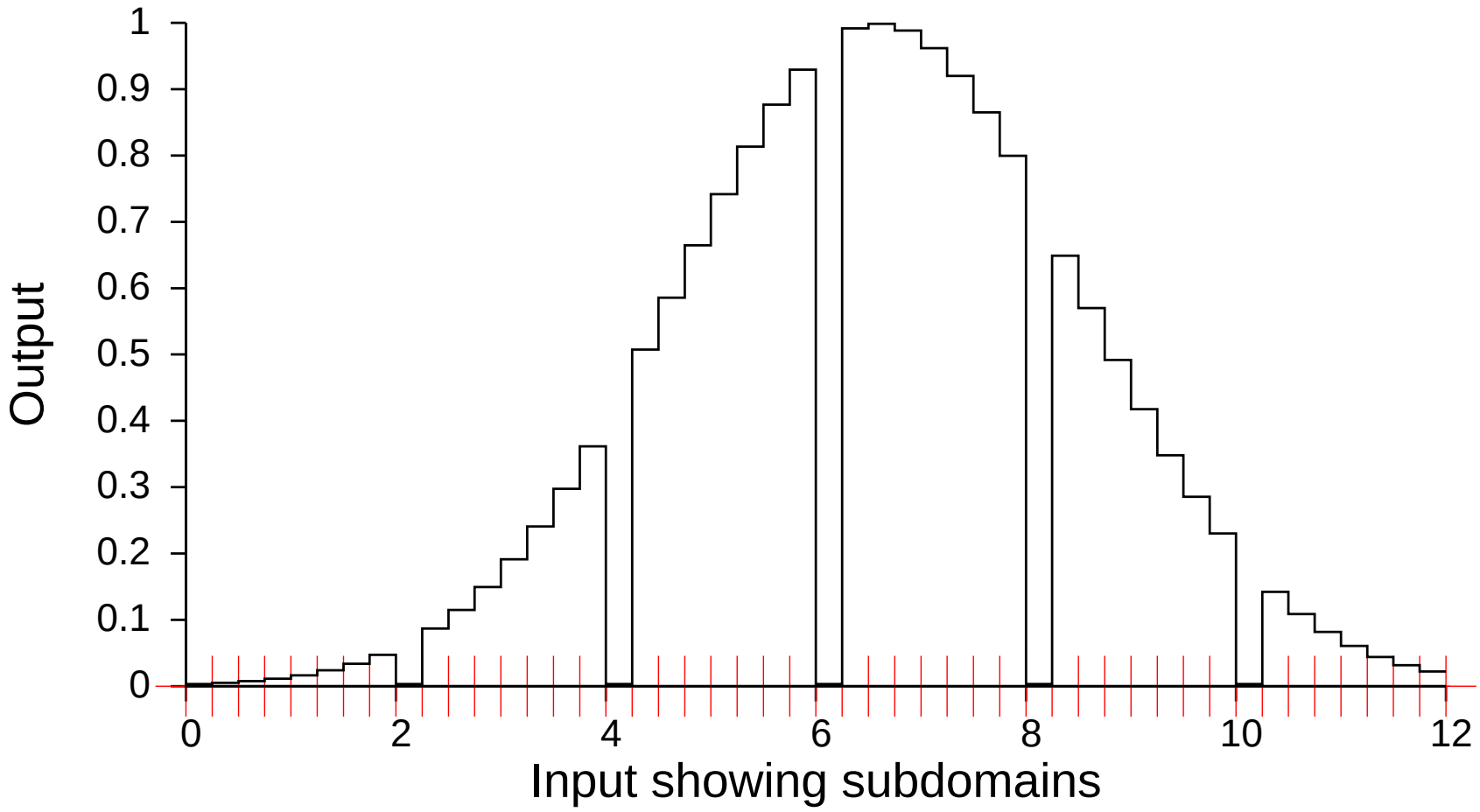
Approximation measurements
(3 samples in each of 48 subdomains):



R-M-S error ranges from 0 to 1.1% by subdomain

Example: `Chop; Bell`

Prediction for system `Chop; Bell`:



Synthesis is fast (5X execution)

R-M-S prediction error 0.1% – 4.5%

Tool-supported Component-based Development

1. Develop components → Repository
   - Write component code
   - *Test code to specification*
   - Choose good subdomains
   - *Approximate with subdomain test*

Key:
(Blue) Human, by-hand *(Red slant) Tools, automatic*

Tool-supported Component-based Development

1. Develop components → Repository
   - Write component code
   - *Test code to specification*
   - Choose good subdomains
   - *Approximate with subdomain test*
2. Design system
   - Describe system and get component approximations
   - *Synthesize system approximation*
   - *Compare with* system specification

Key:
(Blue) Human, by-hand *(Red slant) Tools, automatic*

Tool-supported Component-based Development

1. Develop components → Repository
   - Write component code
   - *Test code to specification*
   - Choose good subdomains
   - *Approximate with subdomain test*
2. Design system
   - Describe system and get component approximations
   - *Synthesize system approximation*
   - *Compare with* system specification
3. Buy components, build system
   - ~~Test system against specification~~

Key:

(Blue) Human, by-hand *(Red slant) Tools, automatic*

Programming in Computer Science Research

These tools were developed over about 10 years.
High school apprentices did a lot of the programming.

The research resulted in the publication of about 6 conference papers, 2 journal papers, and a monograph, about one publication/year – low productivity!

Programming is fun, but don't give up your day job...

More Information

Get the free tools:

```
www.cs.pdx.edu/
~hamlet/
components.html
```

Read the book: