



Executable Formal Semantics for the POSIX Shell

Michael Greenberg *and* Austin J. Blatt

Pomona College

Pomona '18, now Puppet Labs

i'm interested in

powerful

programming languages

you want power?

you want the

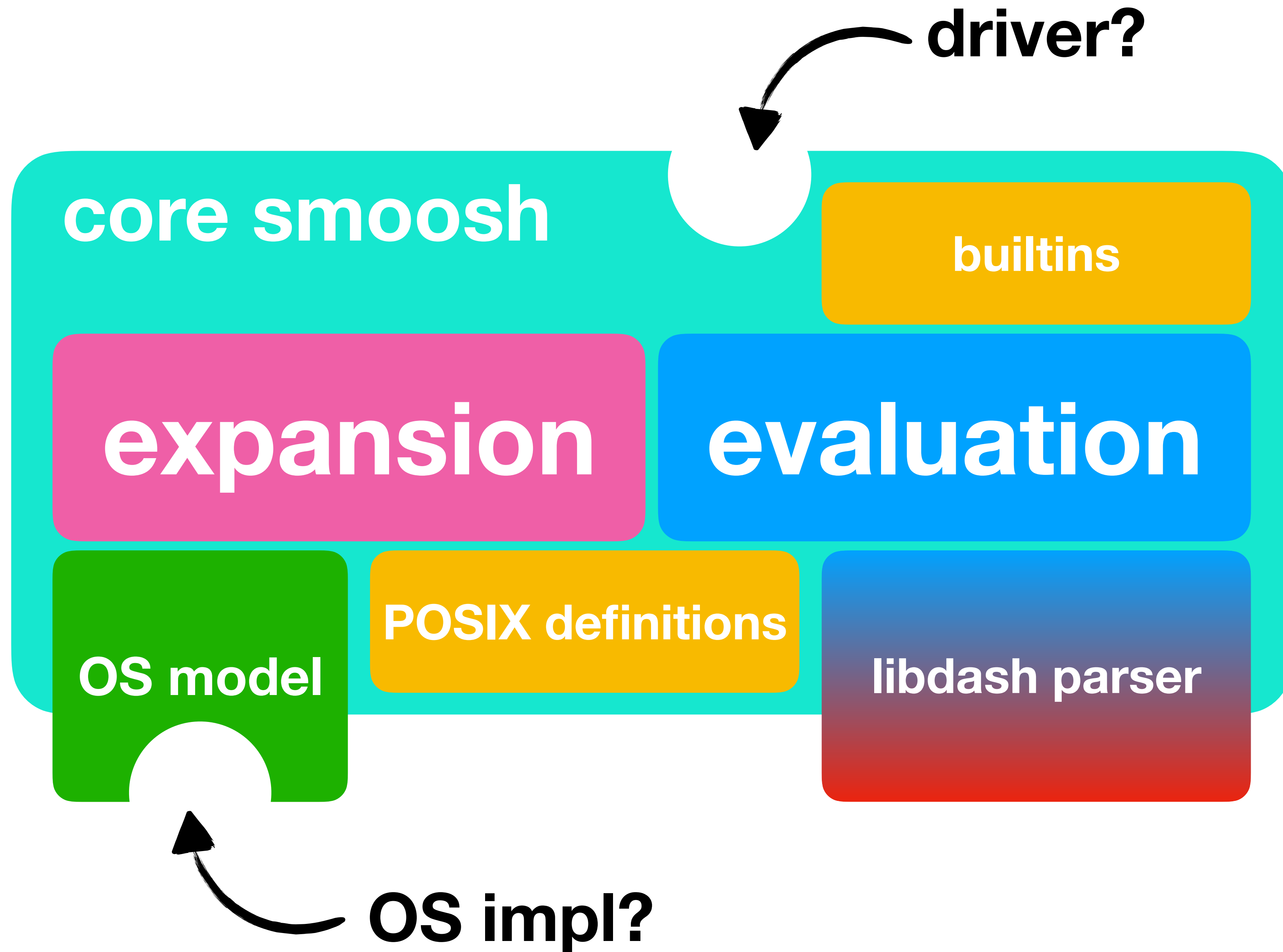
shell



i wrote a new POSIX shell

smooosh

the **S**ymbolic, **M**echanized, **O**bservable, **O**perational **S**hell



system mode

shell driver

core smooosh

builtins

expansion

evaluation

OS model

POSIX definitions

libdash parser

system calls
(OCaml: Sys, Unix, ExtUnix)

symbolic mode

shtepper

core smooosh

builtins

expansion

evaluation

OS model

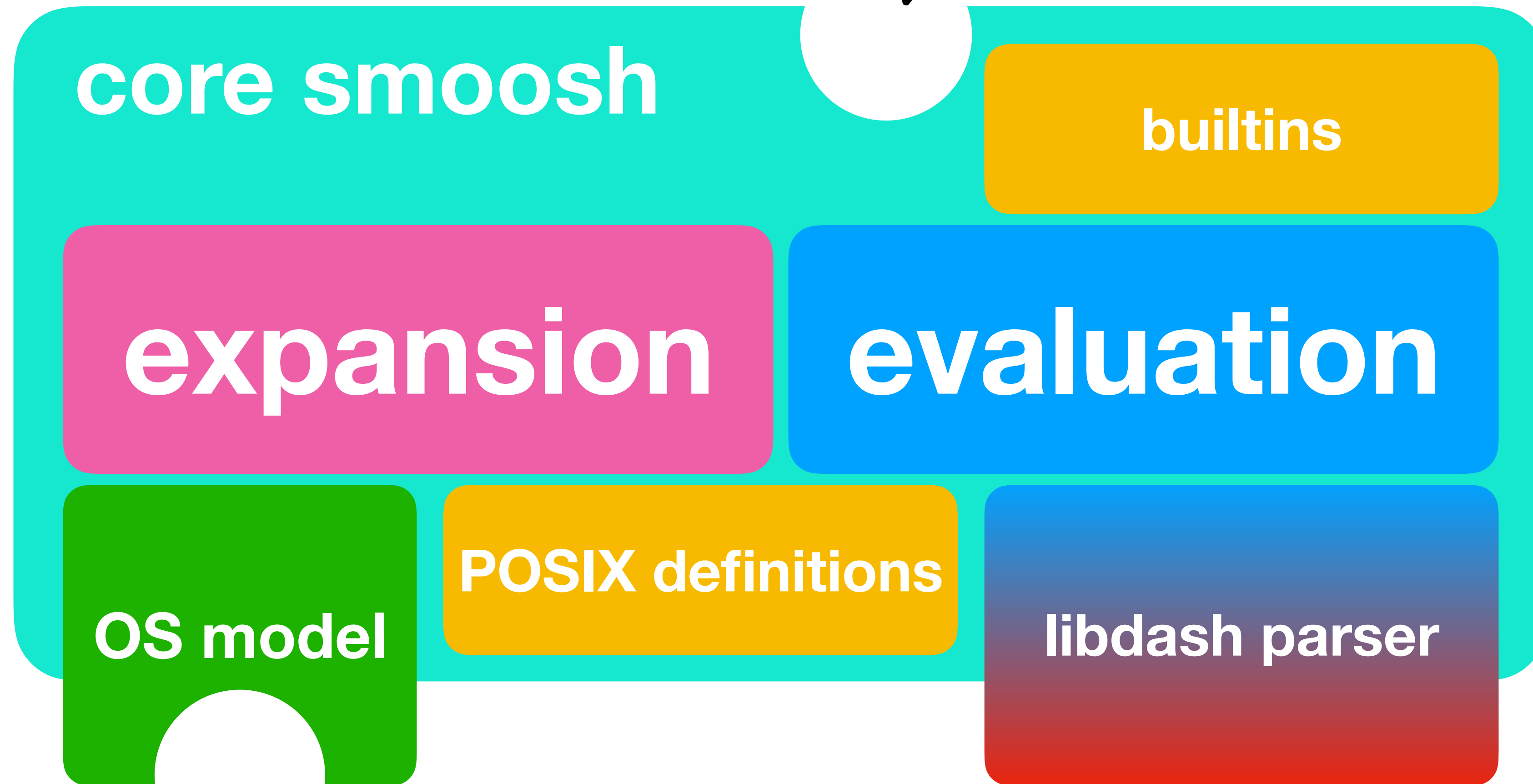
POSIX definitions

libdash parser

POSIX model
(FDs, processes, scheduler, etc.)

your mode

your analysis here?



your model here?

parsing

expansion

evaluation

echo ~ → /Users/mgree

echo \${PWD} → /Users/mgree/talks/smoosh

basename `pwd` → smoosh

echo \$((1+1)) → 2

IFS=""
cat `echo some file` → [shows contents of 'some file']

echo * → abstract.txt posix.key some file

echo you can "" me → you can me

parsing

expansion

evaluation

echo ~ ➔ /Users/mgree

echo \${PWD} ➔ /Users/mgree/talks/smoosh

basename `pwd` ➔ **smoosh**

echo \$((1+1)) ➔ 2

IFS=""
cat `echo some file` ➔ [shows contents of 'some file']

echo * ➔ abstract.txt posix.key some file

echo you can "" me ➔ you can me

```
graph TD; A[parsing] --> B[expansion]; B --> C[evaluation]; C --> A;
```

parsing

expansion

evaluation

`basename `pwd``

```
graph TD; A[parsing] --> B[expansion]; B --> C[evaluation]; C --> A;
```

parsing

expansion

evaluation

basename `pwd`



basename /Users/mgree/talks/smooth

parsing

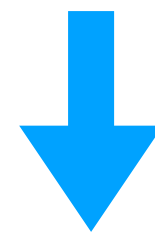
expansion

evaluation

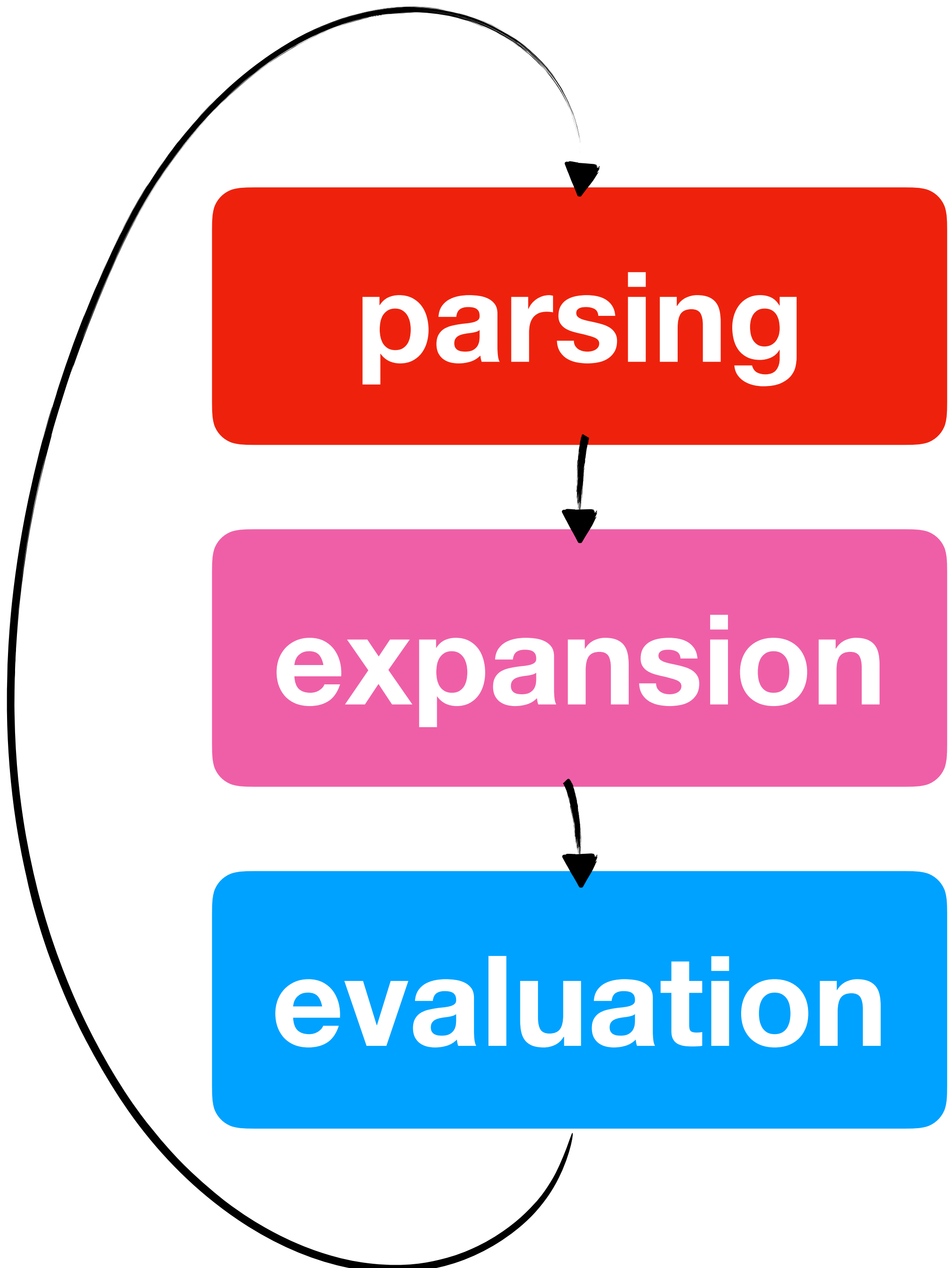
basename `pwd`



basename /Users/mgree/talks/smooth



smooth



parsing

expansion

evaluation

basename `pwd`

expansion

pwd

evaluation

/Users/mgree/talks/smooth

basename /Users/mgree/talks/smooth

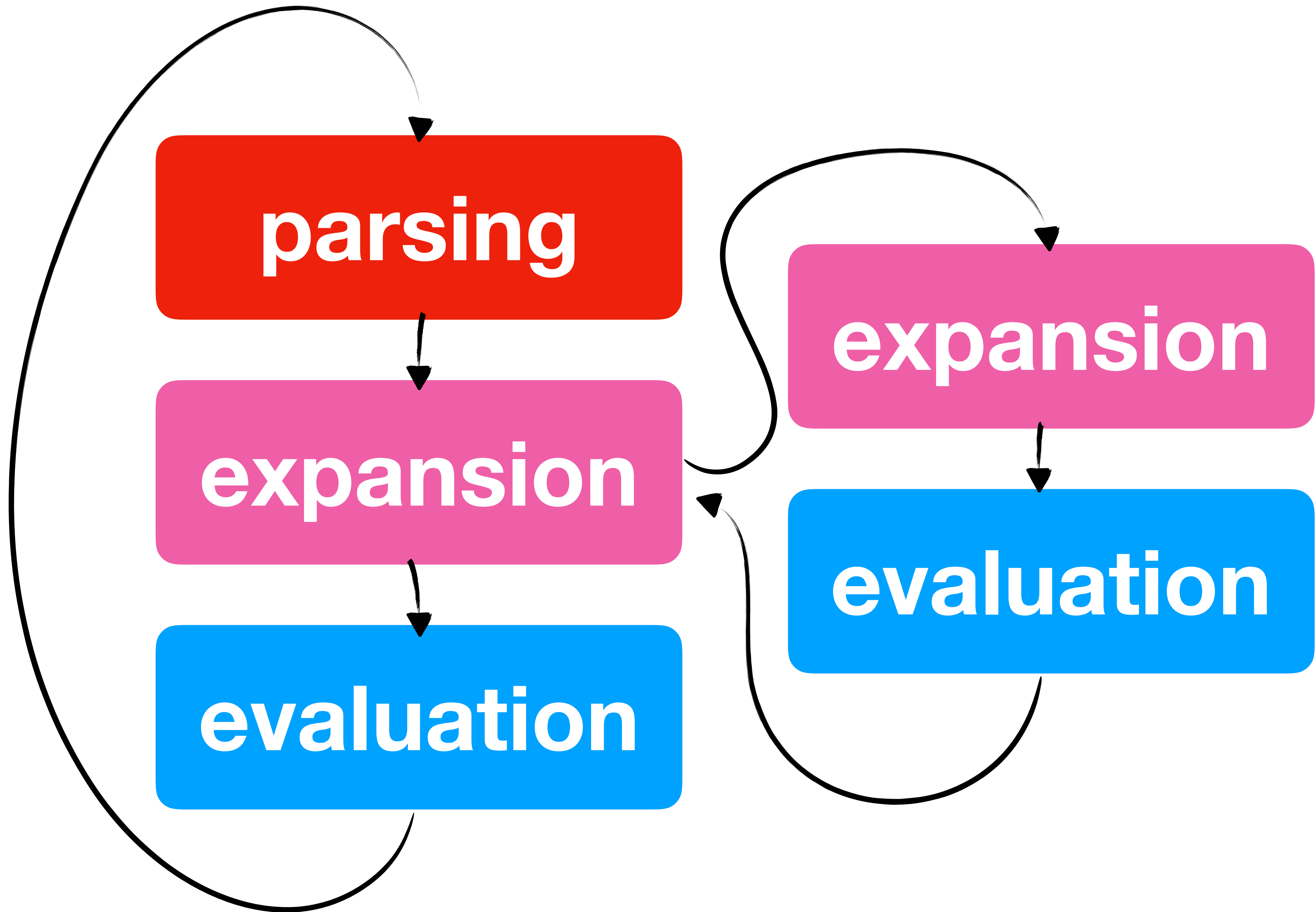
smooth

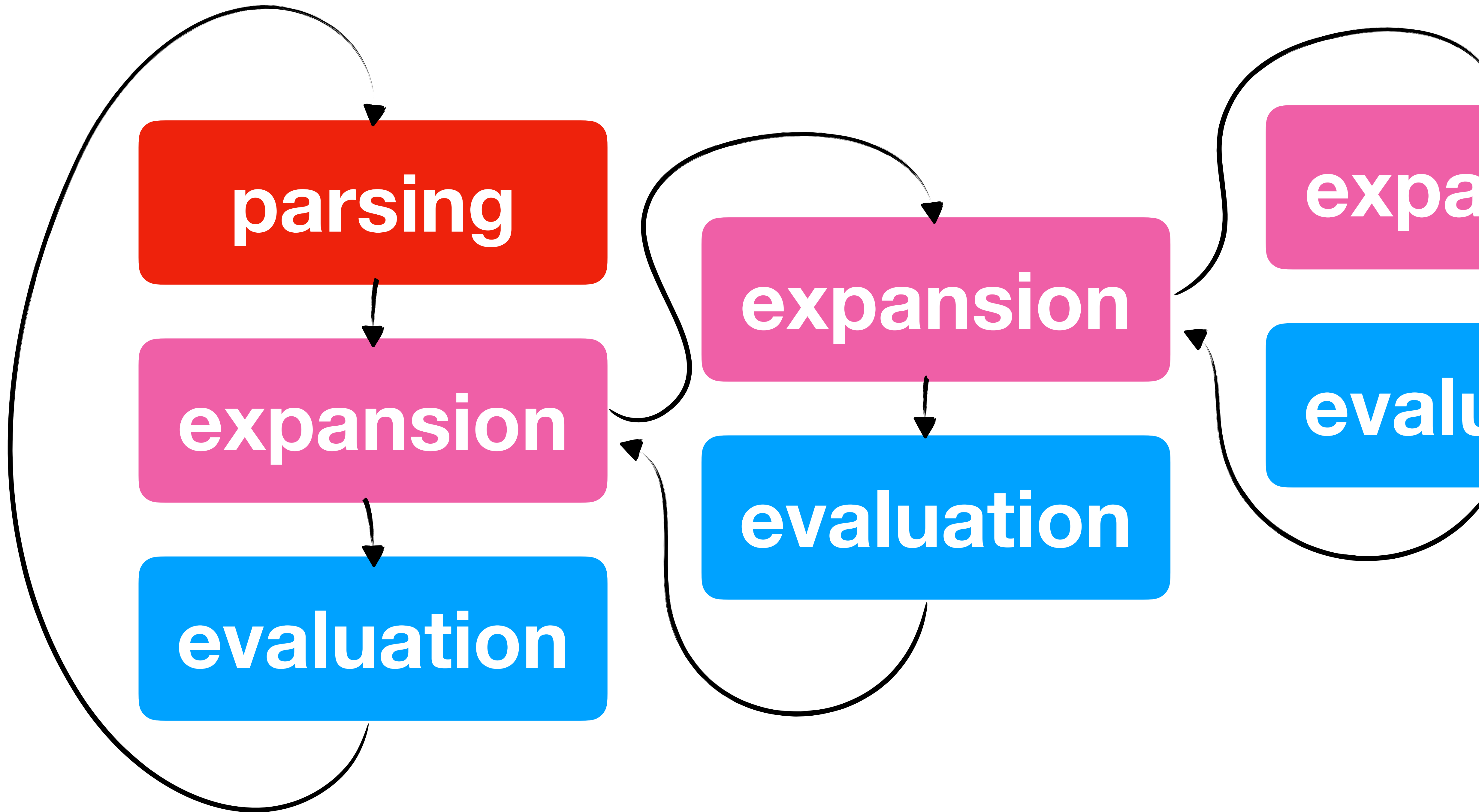
```
graph TD; A[parsing] --> B[expansion]; B --> C[evaluation]; C --> A;
```

parsing

expansion

evaluation





the shtepper

<http://shell.cs.pomona.edu/shtepper>

c ::= v=w ... w ... | c r

| c₁ | c₂ | c₃ | ... | c_n | c & | (c)

| c₁ && c₂ | c₁ || c₂

| ! c | c₁ ; c₂ | if c₁ c₂ c₃

| switch a ... { case w...) c } ...

| while c₁ c₂ | for x in w ... c

| defun v c

c ::= v=w... w ... | c r

| c₁ | c₂ | c₃ | ... | c_n | c & | (c)

| c₁ && c₂ | c₁ || c₂

| ! c | c₁ ; c₂ | if c₁ c₂ c₃

| switch a ... { case w...) c } ...

| while c₁ c₂ | for x in w ... c

| defun v c

c ::= v=w... w ... | c r

| c₁ | c₂ | c₃ | ... | c_n | c & | (c)

| c₁ && c₂ | c₁ || c₂

| ! c | c₁ ; c₂ | if c₁ c₂ c₃

| switch a ... { case w...) c } ...

| while c₁ c₂ | for x in w ... c

| defun v c

Words	$w ::=$	$(s k _)^*$
Control codes	$k ::=$	$\sim \sim s \${s \phi} \$(c) \$((w)) "w"$
Parameter formats	$\phi ::=$	normal length default w ...
Strings	$s \in$	ASCII

C ::= v=w... w ... | c r

| c₁ | c₂ | c₃ | ... | c_n | c & | (c)

| c₁ && c₂ | c₁ || c₂

| ! c | c₁ ; c₂ | if c₁ c₂ c₃

| switch a ... { case w...) c } ...

| while c₁ c₂ | for x in w ... c

| defun v c

Words	$w ::= (s k _)^*$
Control codes	$k ::= \sim \sim s \${s \phi} \$(c) \$((w)) "w"$
Parameter formats	$\phi ::= \text{normal} \text{length} \text{default } w \dots$
Strings	$s \in \text{ASCII}$

: a bunch of spaces
ls ~ / \${x} / * .txt

c ::= v=w... w ... | c r

| c₁ | c₂ | c₃ | ... | c_n | c & | (c)

| c₁ && c₂ | c₁ || c₂

| ! c | c₁ ; c₂ | if c₁ c₂ c₃

| switch a ... { case w...) c } ...

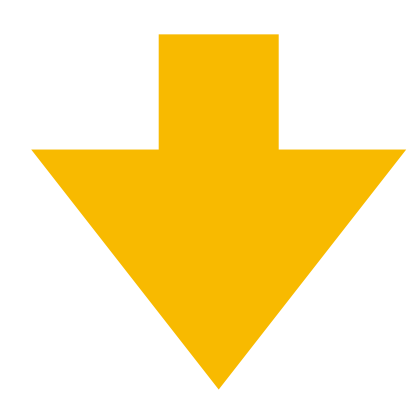
| while c₁ c₂ | for x in w ... c

| defun v c

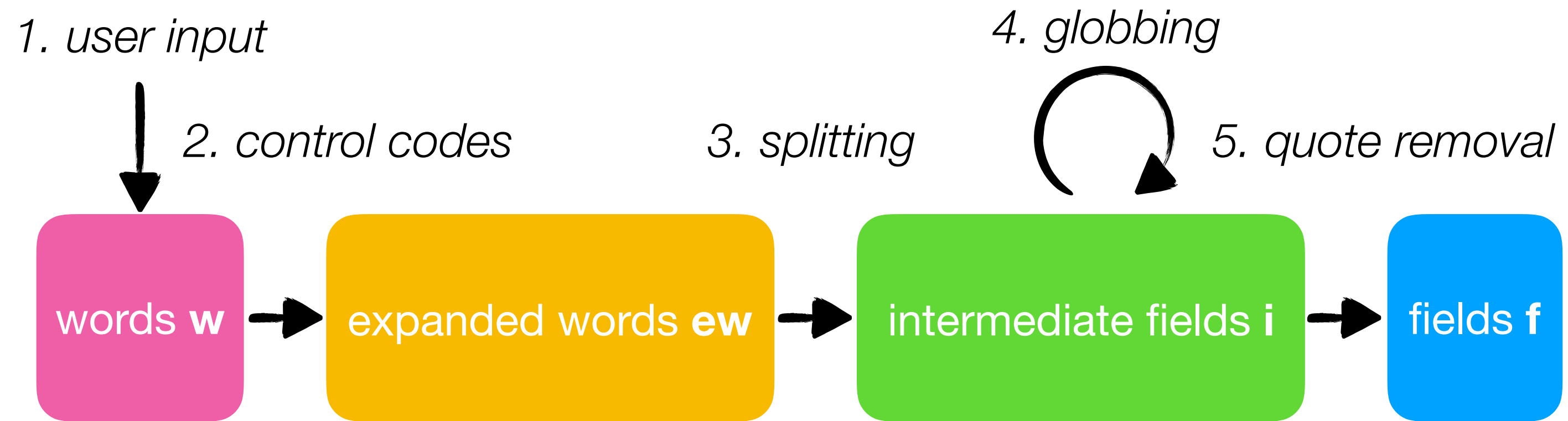
ls **_**
field #1

Words	$w ::= (s k _)^*$
Control codes	$k ::= \sim \sim s \${s \phi} \$(c) \$((w)) "w"$
Parameter formats	$\phi ::= \text{normal} \text{length} \text{default } w \dots$
Strings	$s \in \text{ASCII}$

: a bunch of spaces
ls ~ / \${x} / *.txt



~ / \${x|normal} / *.txt
field #2



Words	$w ::= (s k _)^*$	<i>user input</i> <i>expanded, no splitting or *</i> <i>expanded, split and *</i> <i>fully expanded</i>
Control codes	$k ::= \sim \sim s \${s \phi} \$(c) \$(w) "w"$	
Parameter formats	$\phi ::= \text{normal} \text{length} \text{default } w \dots$	
Strings	$s \in \text{ASCII/locale}$	
Expanded words	$ew ::= (usr s exp s _ @ f "s")^*$	
Intermediate fields	$i ::= (ws _ _ s "s")^*$	
Fields	$f ::= s_1 \dots s_n$	

informative, not in the spec!

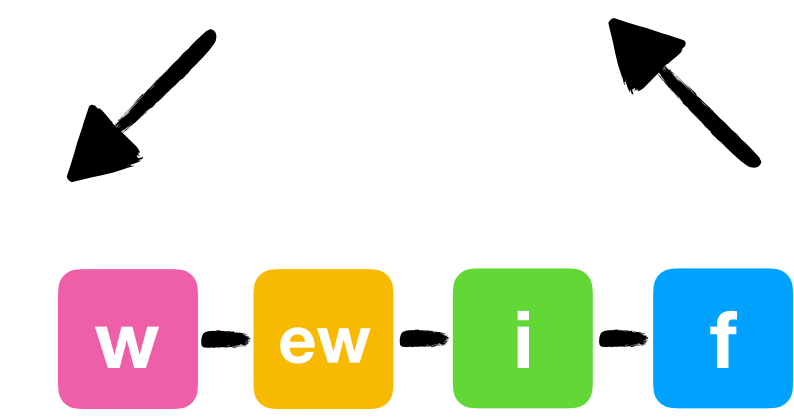
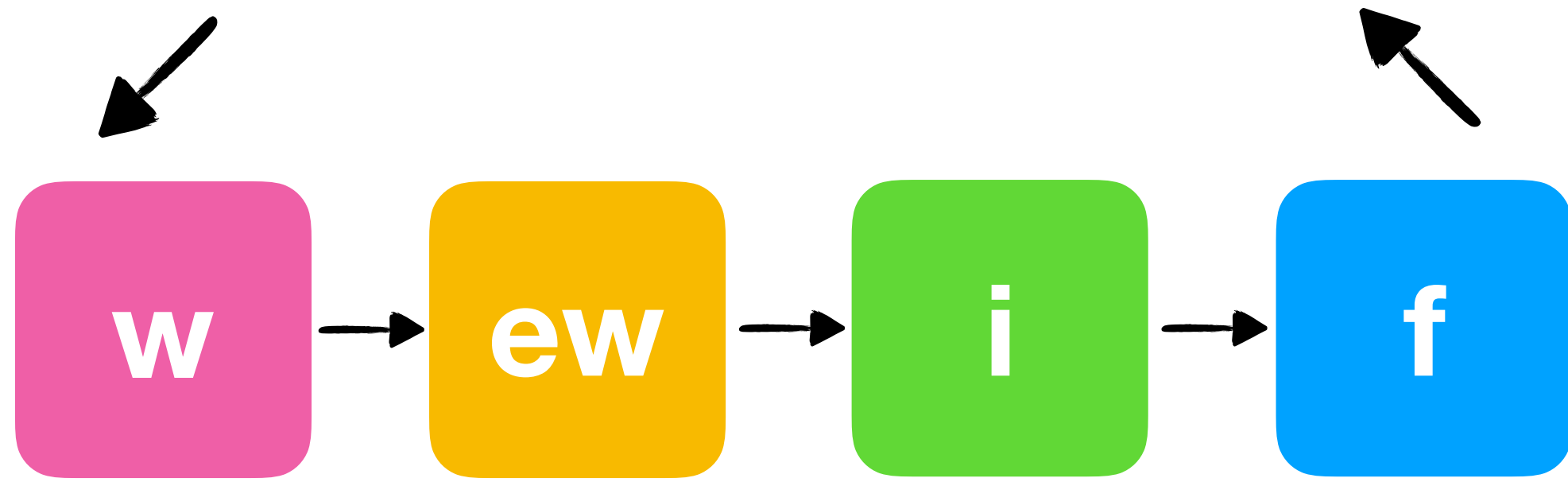
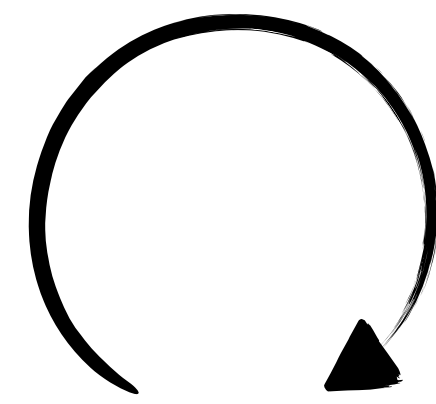
1. user input

2. control codes

3. splitting

4. globbing

5. quote removal



$$w \rightarrow^{*2} ew \rightarrow^{*3} i \rightarrow^{*4,5} f$$

$$\${\mathbf{x=w}} \rightarrow_2 f$$

...

what's unique about the shell's semantics

- Modal: **expansion** and **evaluation**
- Lots of **mutual recursion**
- Language semantics uses **system calls**
- Runtime AST includes nested **command loops**
e.g., `source a/k/a .` and `eval`; top level and `-i`

who cares about semantics?

- **lingua franca** for PL folks
- **symbolic execution**
 - stepper
 - bug finding, static analysis
 - support DevOps tools like Rehearsal
- **correctness** baseline for compilers, other tools
- insights into the **shell** and other **interactive languages**



comparing smoosh

	Smooosh	bash*	dash	zsh [†]	mksh	ksh	yash*
	0.1	4.4-12(1)	0.5.8-2.4	5.3.1-4+b2	54-2+b4	93u+	2.43-1
<i>POSIX test suite (418 tests)</i>							
Failing tests	0	4 (8)	20	×	35	23	22 (23)
Time to run	12m41s	2m43s	2m43s	×	2m52s	3m24s	2m45
<i>Modernish's shell diagnosis (91 potential bugs, 22 potential quirks) and test suite (312 tests)</i>							
Bugs	1	16	2	3	3	14	1
Quirks	0	4	2	5	2	3	8
Failing tests	1	20	3	3	3	17	1
Time to run	5.5s	4.8s	1.4s	1.2s [†]	3.2s	2.2s	2.4s
<i>Smooosh's test suite (161 tests)</i>							
Failing tests	0	30 (35)	42	52	34	41	43 (39)
Time to run	23s	28s	1m13s	42s	21s	28s	29s

comparing smoosh

	Smooosh	bash*	dash	zsh [†]	mksh	ksh	yash*
	0.1	4.4-12(1)	0.5.8-2.4	5.3.1-4+b2	54-2+b4	93u+	2.43-1
<i>POSIX test suite (418 tests)</i>							
Failing tests	0	4 (8)	20	×	35	23	22 (23)
Time to run	12m41s	2m43s	2m43s	×	2m52s	3m24s	2m45
<i>Modernish's shell diagnosis (91 potential bugs, 22 potential quirks) and test suite (312 tests)</i>							
Bugs	1	16	2	3	3	14	1
Quirks	0	4	2	5	2	3	8
Failing tests	1	20	3	3	3	17	1
Time to run	5.5s	4.8s	1.4s	1.2s [†]	3.2s	2.2s	2.4s
<i>Smooosh's test suite (161 tests)</i>							
Failing tests	0	30 (35)	42	52	34	41	43 (39)
Time to run	23s	28s	1m13s	42s	21s	28s	29s

comparing smooosh

	Smooosh	bash*	dash	zsh [†]	mksh	ksh	yash*
	0.1	4.4-12(1)	0.5.8-2.4	5.3.1-4+b2	54-2+b4	93u+	2.43-1
<i>POSIX test suite (418 tests)</i>							
Failing tests	0	4 (8)	20	×	35	23	22 (23)
Time to run	12m41s	2m43s	2m43s	×	2m52s	3m24s	2m45
<i>Modernish's shell diagnosis (91 potential bugs, 22 potential quirks) and test suite (312 tests)</i>							
Bugs	1	16	2	3	3	14	1
Quirks	0	4	2	5	2	3	8
Failing tests	1	20	3	3	3	17	1
Time to run	5.5s	4.8s	1.4s	1.2s [†]	3.2s	2.2s	2.4s
<i>Smooosh's test suite (161 tests)</i>							
Failing tests	0	30 (35)	42	52	34	41	43 (39)
Time to run	23s	28s	1m13s	42s	21s	28s	29s

who cares about semantics?

- 10 **bugs** in the POSIX test suite
- 3 **patches** for dash
- dozens of other, **uninvestigated** bugs



POSIX test suite bug: impatient pipes

```
(trap "" PIPE
```

```
sleep 2
```

```
echo "it's alive" >out
```

```
) | true
```

```
[ $(cat out) = "it's alive" ]
```

POSIX test suite bug: impatient pipes

```
(trap "" PIPE  
sleep 2  
echo "it's alive" >out  
) | true  
[ $(cat out) = "it's alive" ]
```

If the pipeline is not in the background (see *Asynchronous Lists*), the shell shall wait for the last command specified in the pipeline to complete, and **may** also wait for all commands to complete.

IEEE Std 1003.1-2017 §2.9.2

dash bug: unset arithmetic

```
unset x
```

```
[  $$(x + 2)$  -eq 2 ]
```



dash bug: unset arithmetic

unset x

```
[ $((x + 2)) -eq 2 ]
```



dash: 1: Illegal number:



my motivation



“So, what do you do?”

my motivation

THE BREAKTHROUGH FRANKLIN PC-8000

A new-generation home or office computer so affordable, anyone can join the computer revolution

Simply plug it in —
It's all there!

Check how many "add-ons" are built-in at **NO EXTRA COST!**

IBM PC compatible
Runs IBM PC compatible software.

Powerful **512K RAM** expandable to 640K

Color Graphics Card with RGB, composite color & composite monochrome

MS-DOS 3.1 disk included

2 floppy disk drives each with 360K storage capacity

Power supply supports 10 & 20 megabyte Seagate hard disk drives

Parallel Port for connecting printer

Serial (RS-232) Port for connecting modem, networking or other serial devices

Game Port

Expansion — standard IBM PC bus for adding hard disk, modem, or other accessory cards.

Numeric keypad with cursor control

Detachable keyboard with 10 programmable function keys and dedicated editing keys

It might sound too good to be true... if it wasn't a FRANKLIN.



What happened?

smoosh

~11k loc	shell semantics	Lem, OCaml	~5k loc	AST, OS model
~1k loc	testing	OCaml	~3k loc	expansion and evaluation
~0.5 loc	dash bindings	C, OCaml	~2.5k loc	builtin commands

- **small-step** operational semantics
 - **expansion** and **evaluation** are mutually recursive
- semantics uses an **abstract OS interface**
 - **system** (real, working shell) or **symbolic** (fake OS)
- **1.5 person years** of effort

