# FSharp.Data.Adaptive

## Taming Mutation

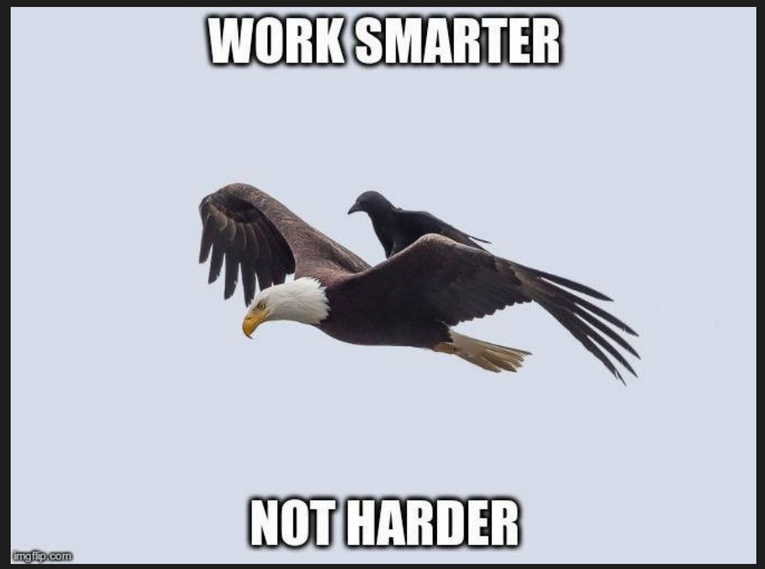
**Georg Haaser 26.06.2023**

https://github.com/fsprojects/FSharp.Data.Adaptive

# What is Incremental Computation?

- Compute only what is needed as inputs change
- Track dependencies in code
- Re-execute affected parts when changes happen

```
main: main.o lib.o
	gcc -o main main.o lib.o

main.o: main.c
	gcc -c main.c

lib.o: lib.c
	gcc -c lib.c
```

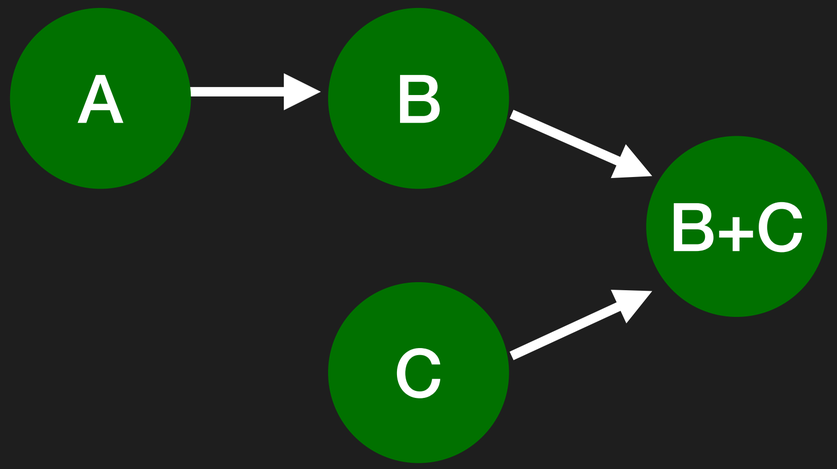



| | B | C |
|---|---|---|
| 1 | 3 | |
| 2 | 4 | =B1+B2 |
| 3 | | |
| 4 | | |

# As a Library

- Container-Cells for changeable values holding dependency information
- internally track in-/outputs for each cell
- Change-propagation mechanism for marking cells as dirty

```
type aval<'a> =
    abstract Dirty : IEvent<unit>
    abstract GetValue : unit -> 'a
```




Self Adjusting Computation [Acar 2005] https://drive.google.com/file/d/19UcnvDS1_6opK5qZcceuDjHTLmG_9Ovf/view

```fsharp
type aval<'a> =
    abstract Dirty : IEvent<unit>
    abstract GetValue : unit -> 'a

type cval<'a> =
    interface aval<'a>
    new : 'a -> cval<'a>
    abstract Value : 'a with get, set

module AVal =
    val constant : 'a -> aval<'a>
    val map : ('a -> 'b) -> aval<'a> -> aval<'b>
    val bind : ('a -> aval<'b>) -> aval<'a> -> aval<'b>
```

# Demo

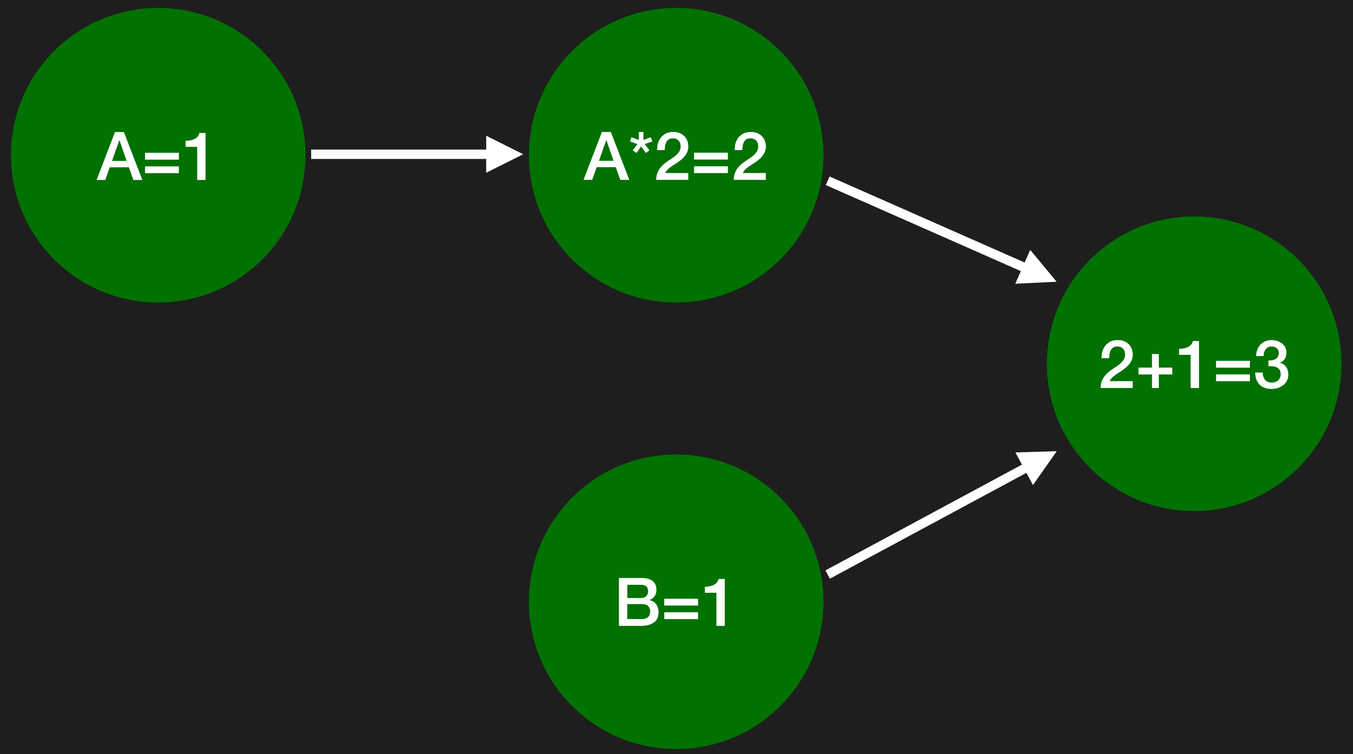
https://github.com/krauthaufen/AdaptiveDemo

# What for ?

- User Interfaces https://github.com/krauthaufen/Fable.Elmish.Adaptive
- Realtime Graphics https://github.com/aardvark-platform/aardvark.rendering
- IDE tooling https://github.com/fsharp/FsAutoComplete/
- Database views
- Basically everything that needs to adapt to inputs

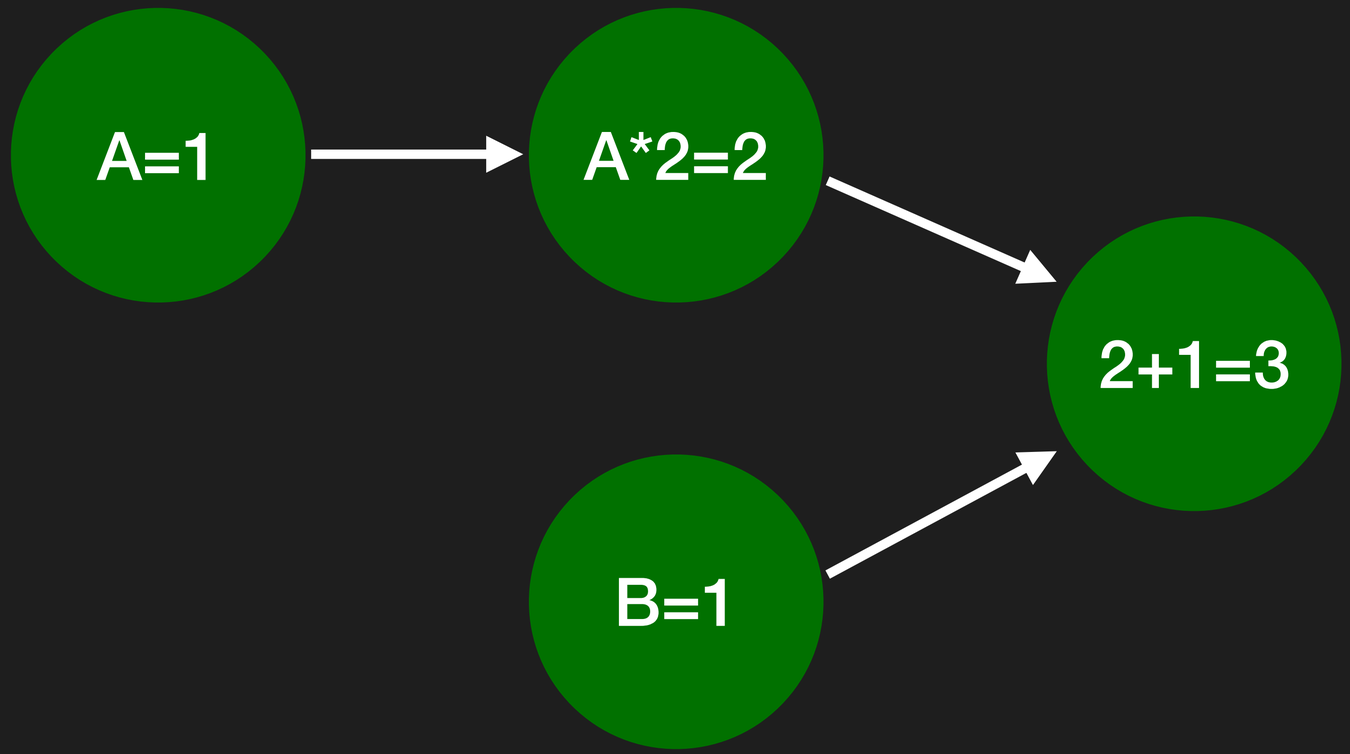
# How?

```
let a = cval 1
let b = cval 1

let c = a |> AVal.map ((*) 2)
let d = (c, b) ||> AVal.map2 (+)
```



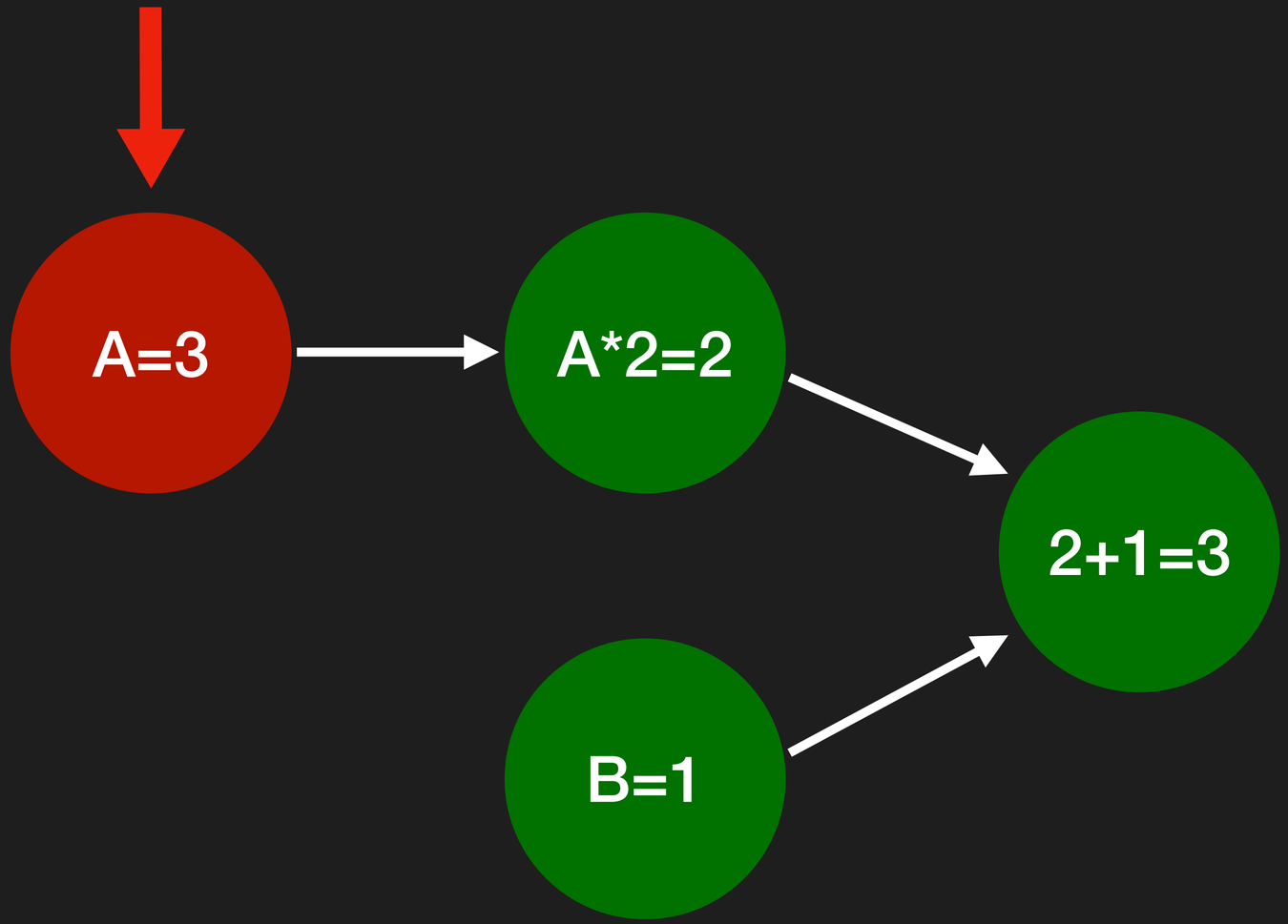
# How?

```
let a = cval 1
let b = cval 1

let c = a |> AVal.map ((*) 2)
let d = (c, b) ||> AVal.map2 (+)

transact (fun () ->
    a.Value <- 3
)
```



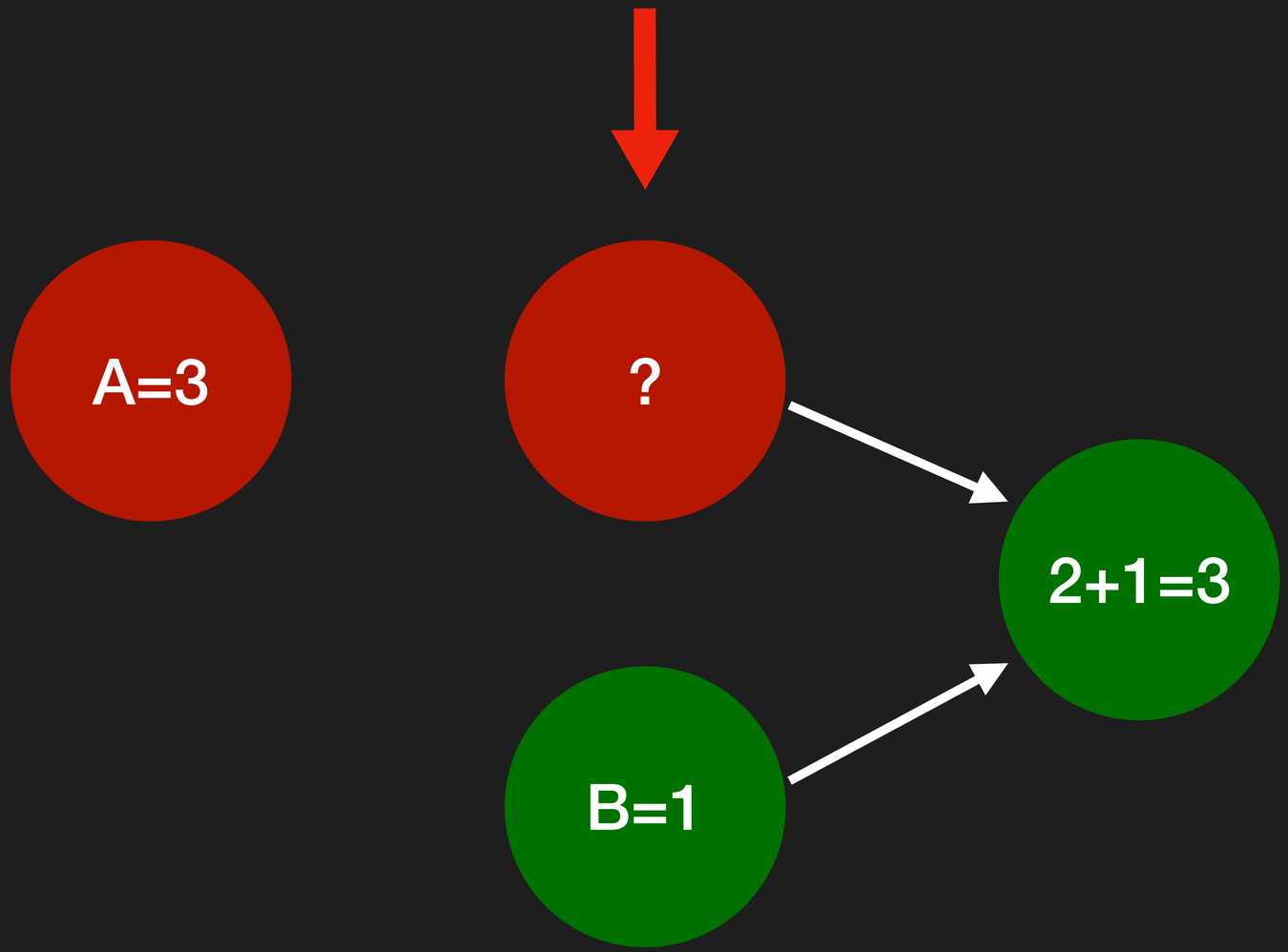
# How?

```
let a = cval 1
let b = cval 1

let c = a |> AVal.map ((*) 2)
let d = (c, b) ||> AVal.map2 (+)

transact (fun () ->
    a.Value <- 3
)
```



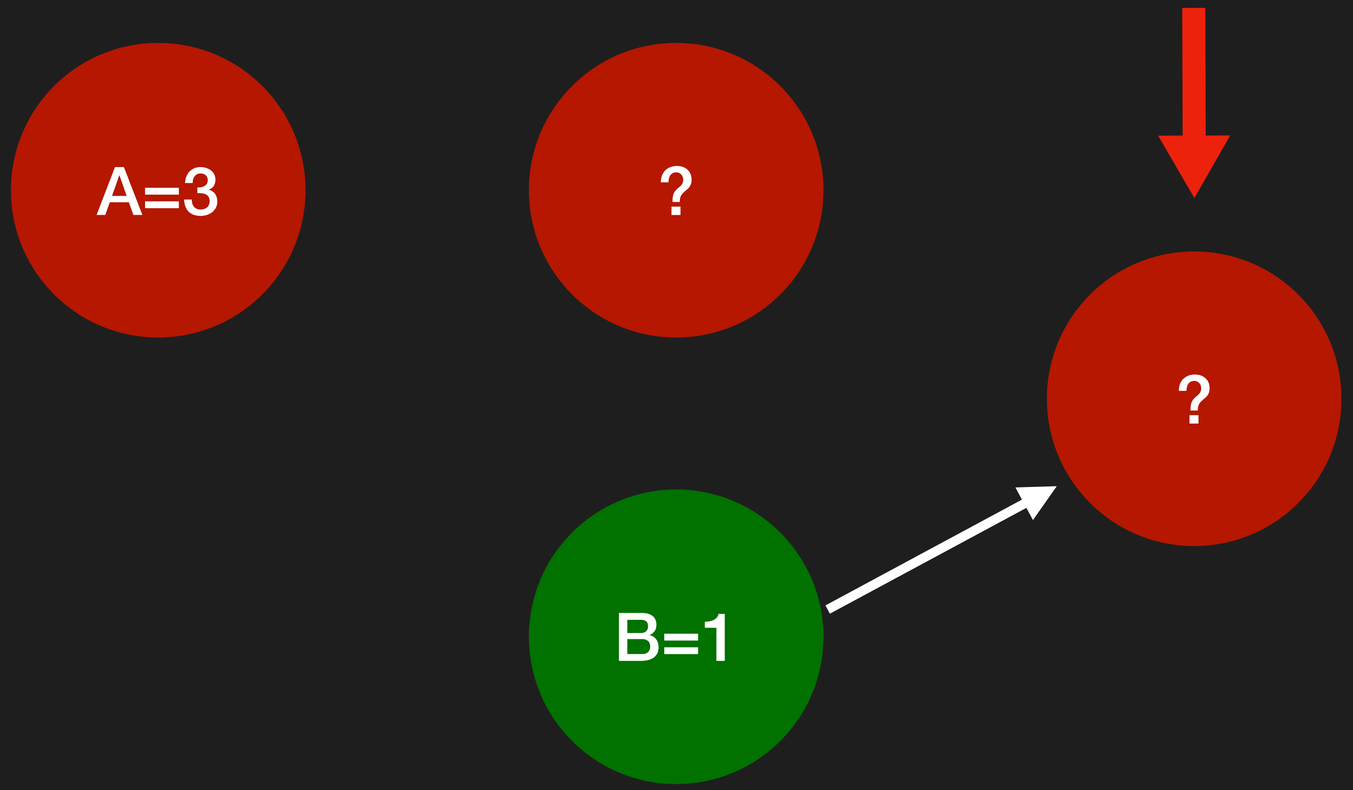
# How?

```
let a = cval 1
let b = cval 1

let c = a |> AVal.map ((*) 2)
let d = (c, b) ||> AVal.map2 (+)

transact (fun () ->
    a.Value <- 3
)
```



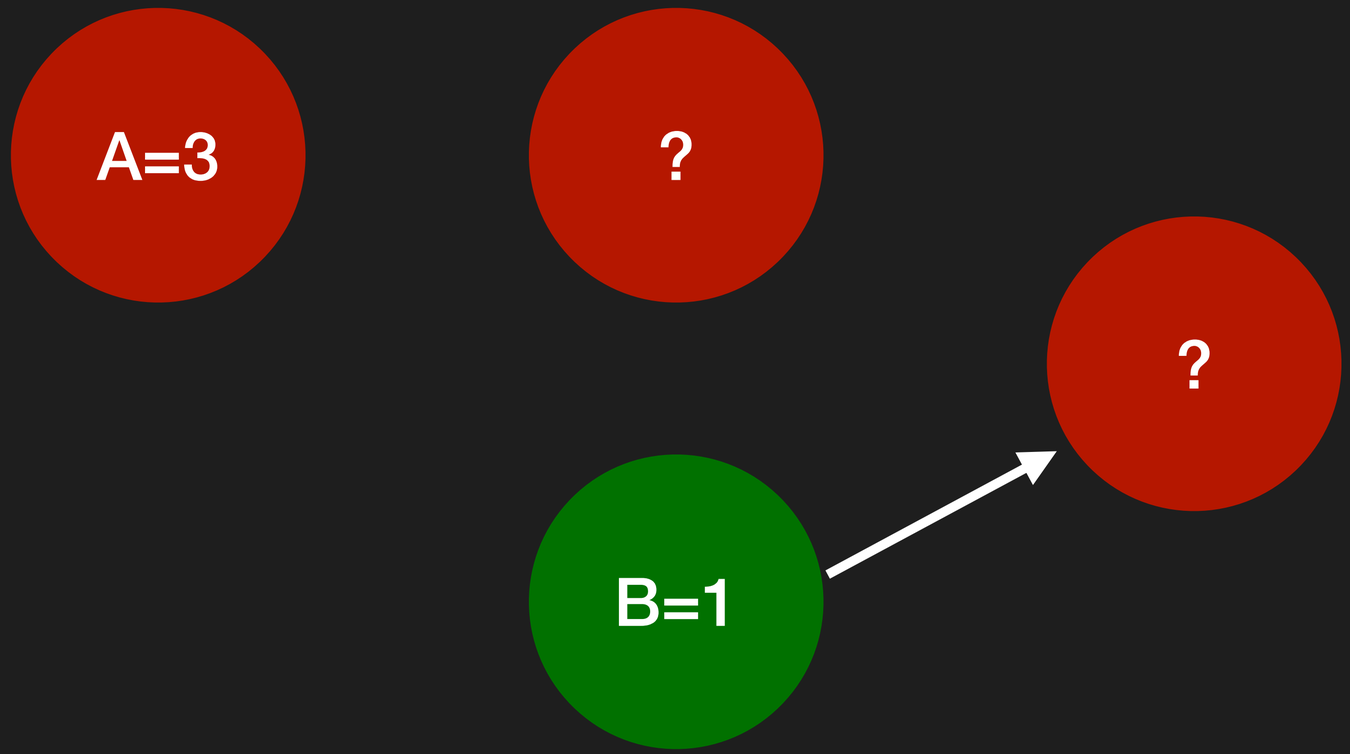
# How?

```
let a = cval 1
let b = cval 1

let c = a |> AVal.map ((*) 2)
let d = (c, b) ||> AVal.map2 (+)

transact (fun () ->
    a.Value <- 3
)
```



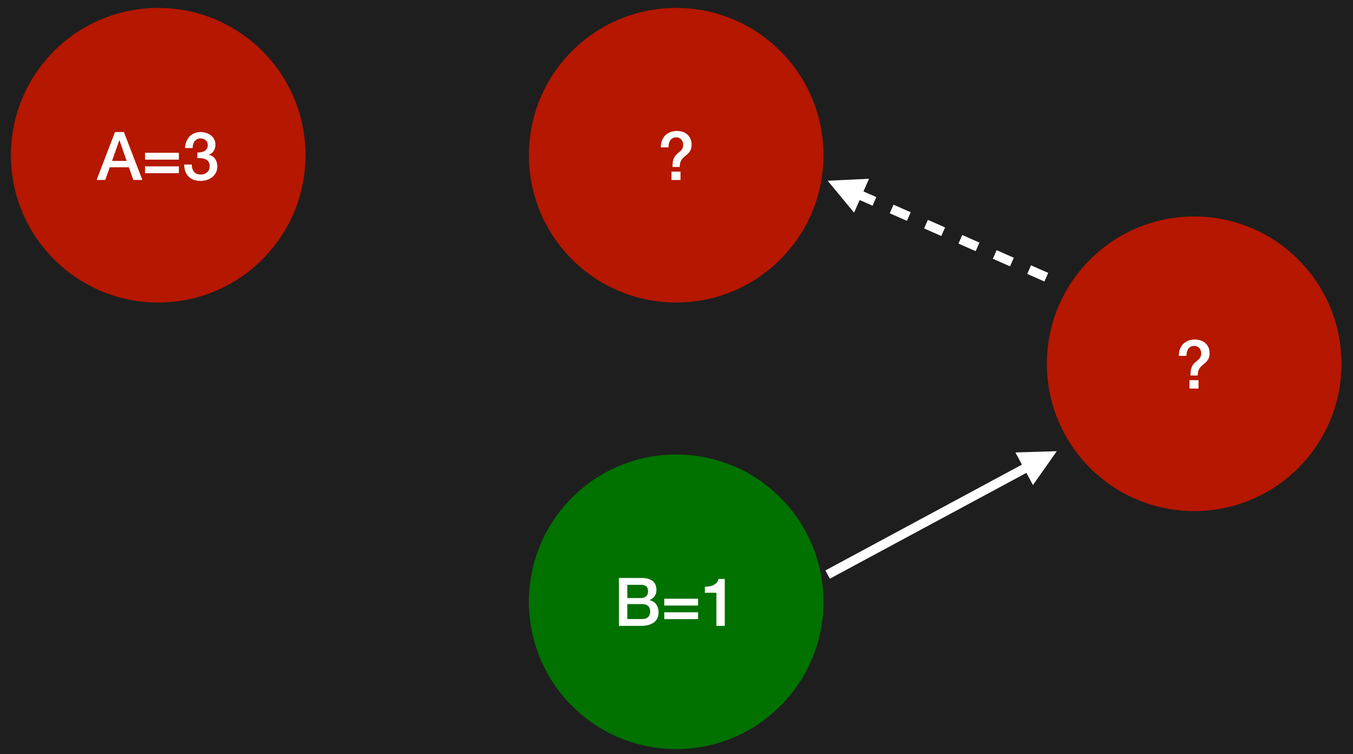
# How?

```
let a = cval 1
let b = cval 1

let c = a |> AVal.map ((*) 2)
let d = (c, b) ||> AVal.map2 (+)

transact (fun () ->
    a.Value <- 3
)
```



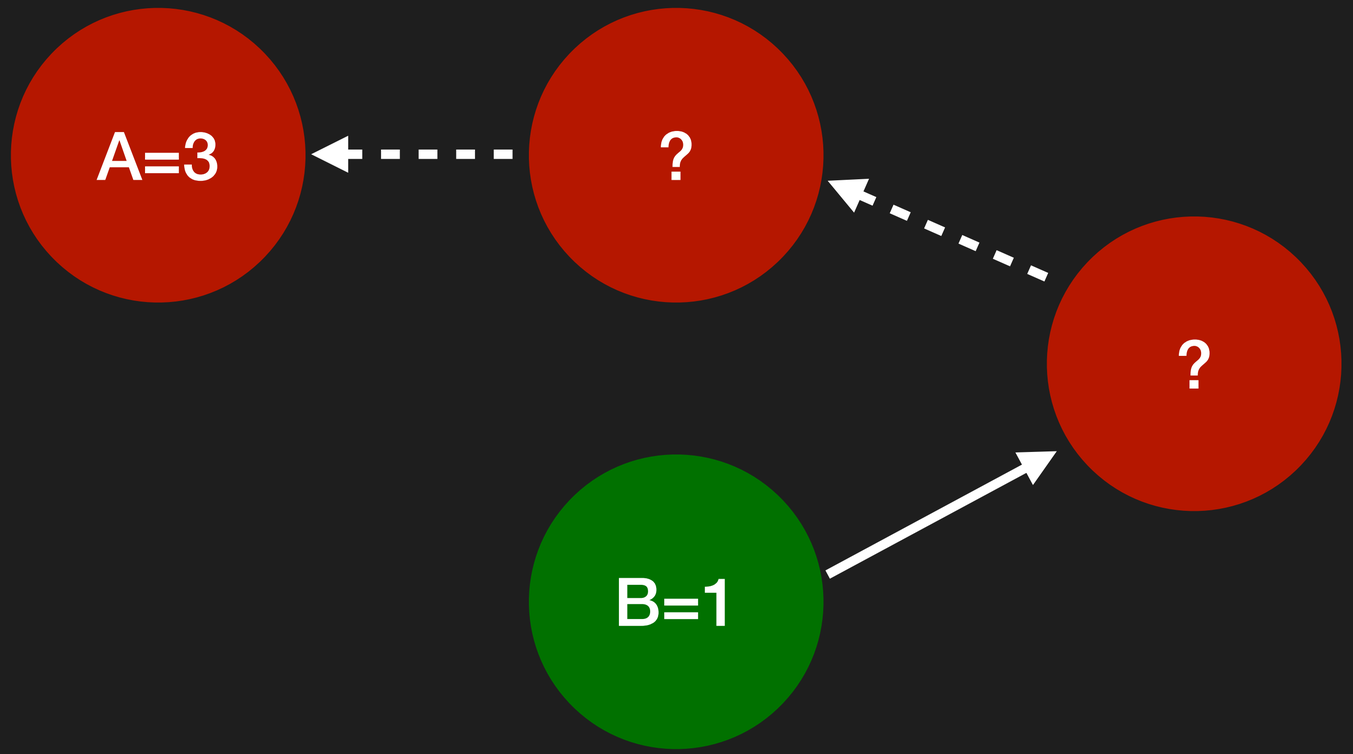
# How?

```fsharp
let a = cval 1
let b = cval 1

let c = a |> AVal.map ((*) 2)
let d = (c, b) ||> AVal.map2 (+)

transact (fun () ->
    a.Value <- 3
)

AVal.force d
```



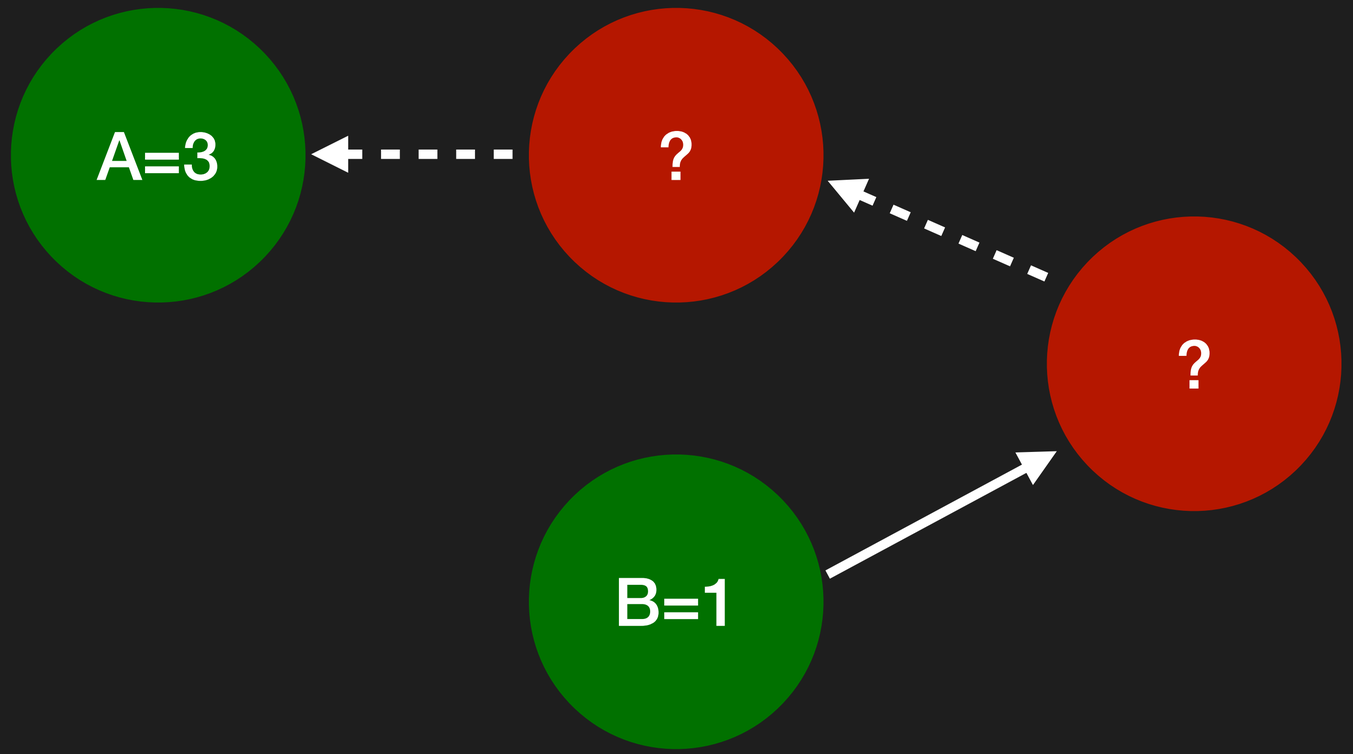
# How?

```
let a = cval 1
let b = cval 1

let c = a |> AVal.map ((*) 2)
let d = (c, b) ||> AVal.map2 (+)

transact (fun () ->
    a.Value <- 3
)

AVal.force d
```



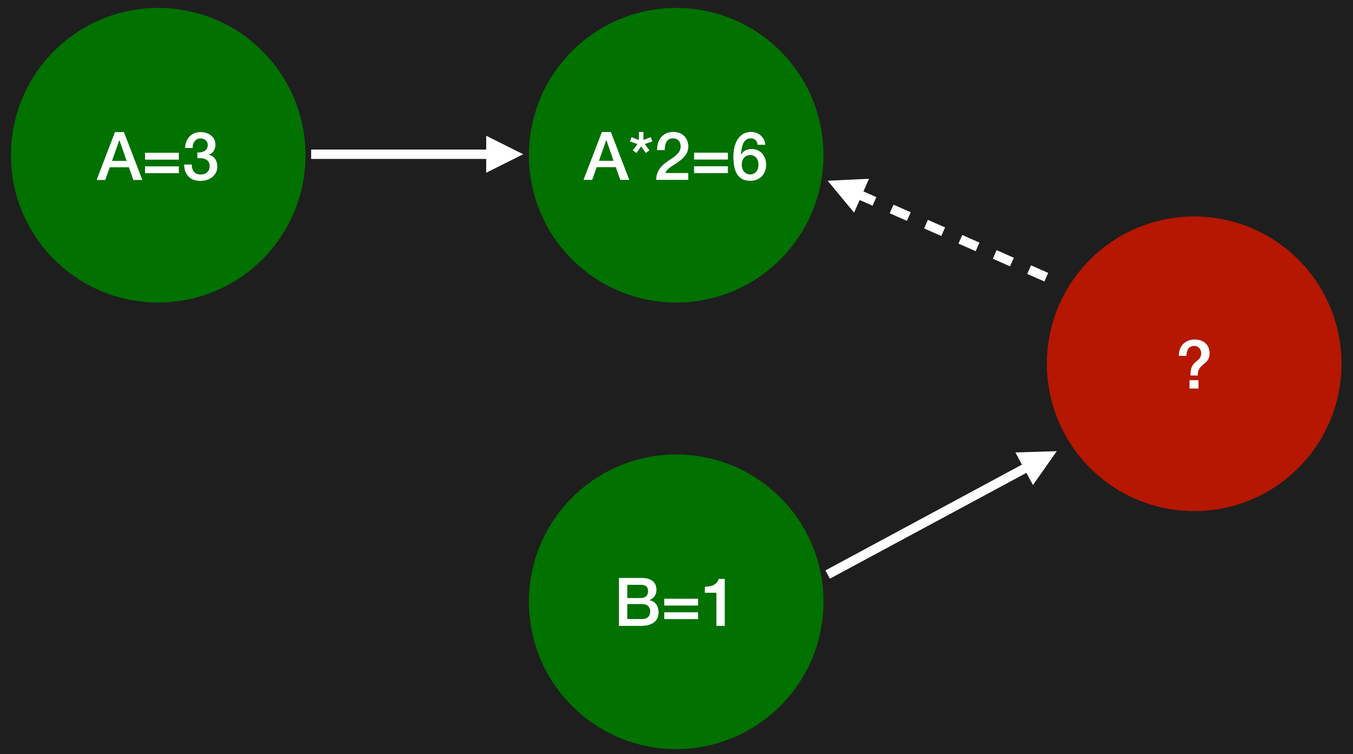
# How?

```
let a = cval 1
let b = cval 1

let c = a |> AVal.map ((*) 2)
let d = (c, b) ||> AVal.map2 (+)

transact (fun () ->
    a.Value <- 3
)

AVal.force d
```



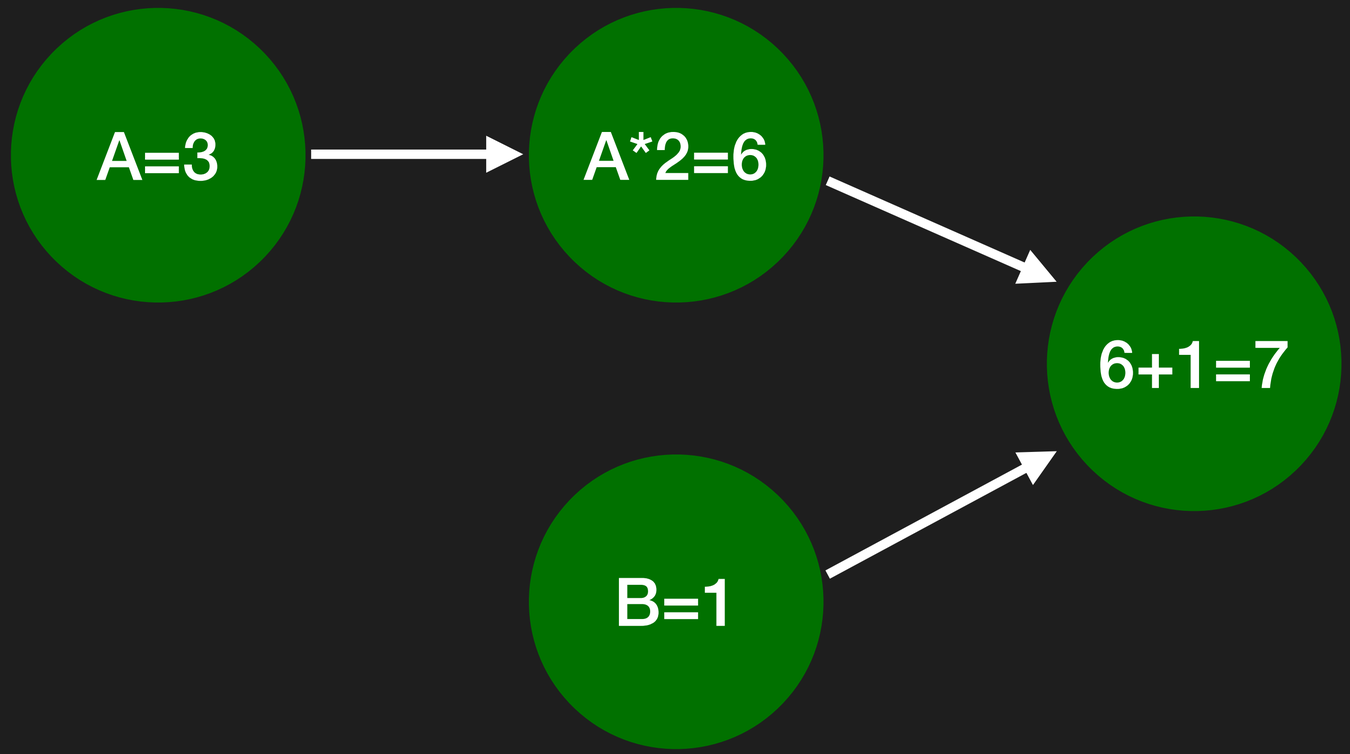
# How?

```
let a = cval 1
let b = cval 1

let c = a |> AVal.map ((*) 2)
let d = (c, b) ||> AVal.map2 (+)

transact (fun () ->
    a.Value <- 3
)

AVal.force d
```



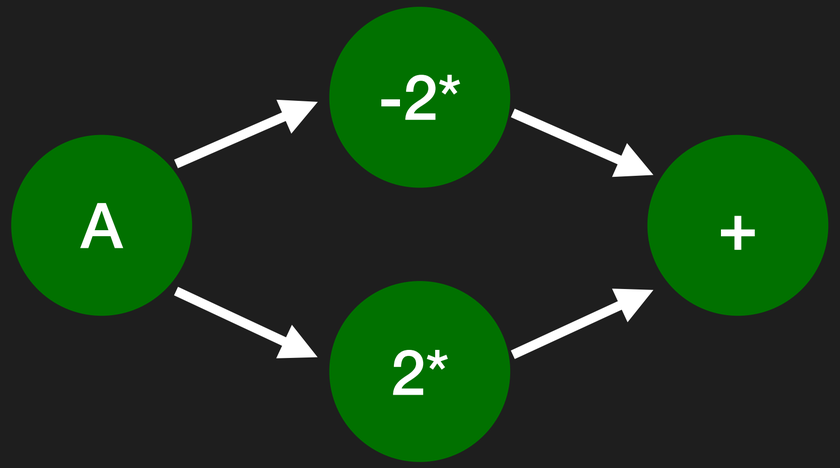
# How?

```
let a = cval 1
let b = cval 1

let c = a |> AVal.map ((*) 2)
let d = (c, b) ||> AVal.map2 (+)

transact (fun () ->
    a.Value <- 3
)

AVal.force d
```



# Implementation

- Values, Sets, Lists and Maps
- Well documented & tested
- Thread safe
- Eventually consistent
- No conceptual memory leaks
- Works with Fable Compiler
- Experimental Task/Async interop for FSAC with cancellation



# Application Demo

https://github.com/krauthaufen/AdaptiveDemo

# Get in Touch

https://github.com/fsprojects/FSharp.Data.Adaptive

https://discord.gg/DEcGCpT2

georg.haaser@aardworx.at

# Thank You