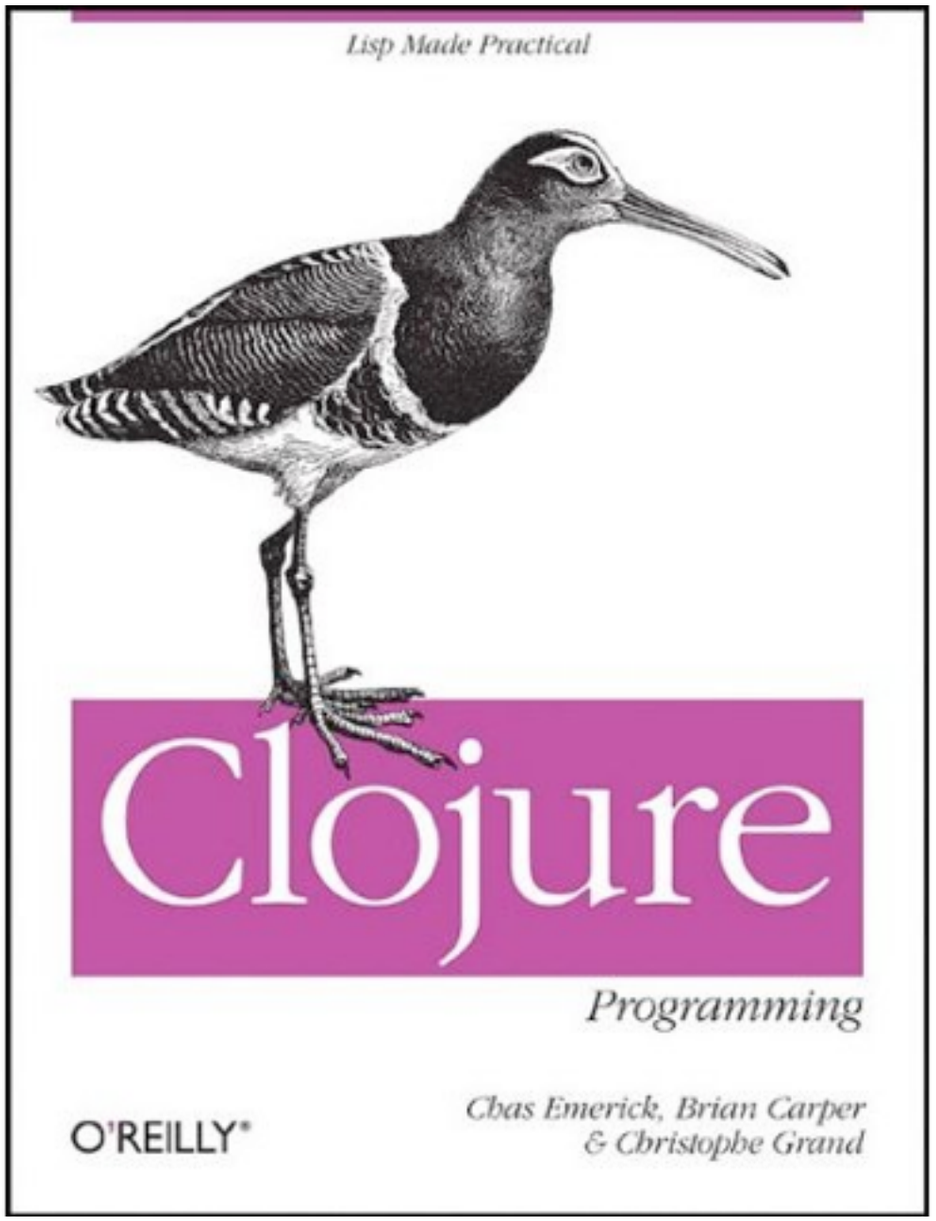
Modeling the world probabilistically using Bayesian networks in Clojure

Chas Emerick
@cemerick
http://cemerick.com
http://snowtide.com

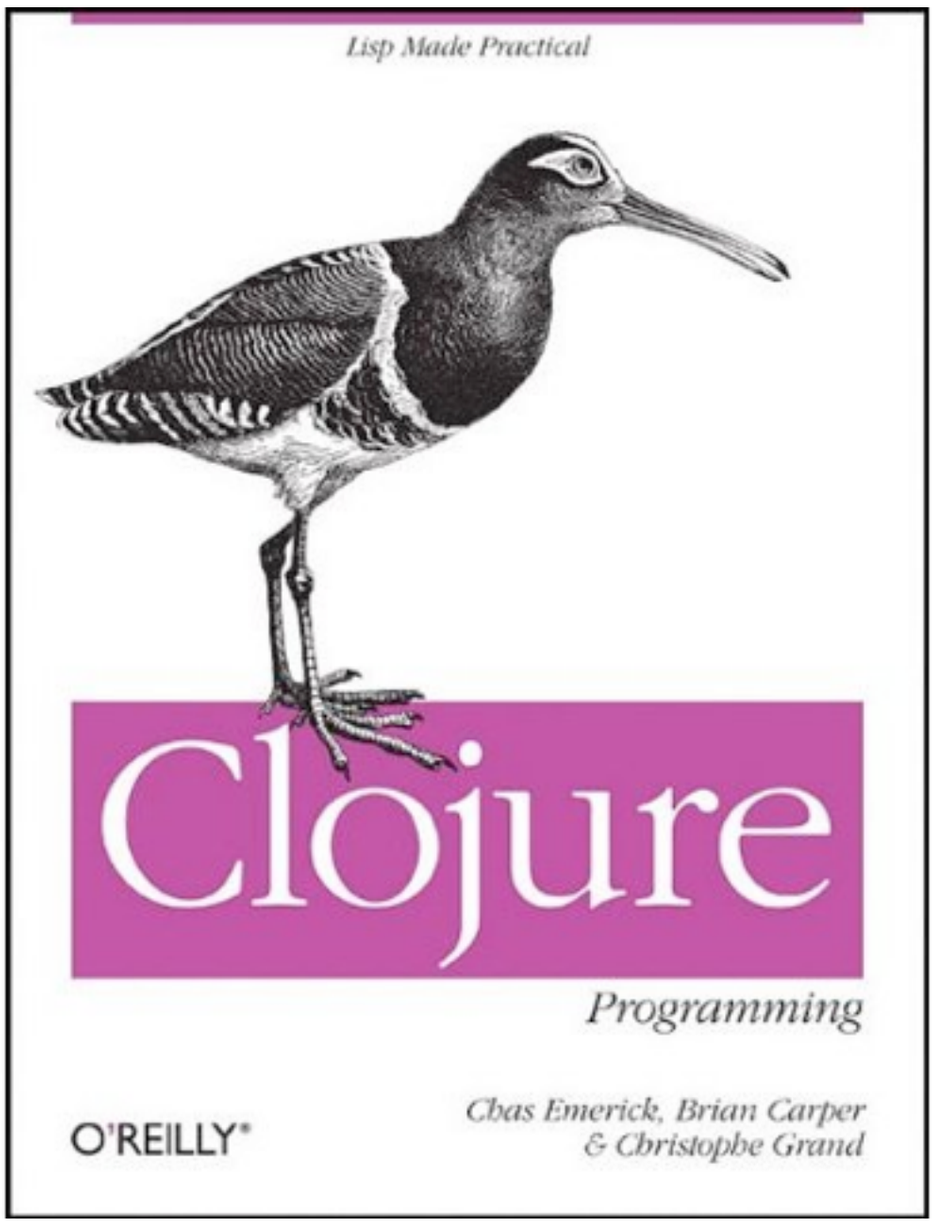
@cemerick



Snowtide

@cemerick

Clojure full-time since 2008

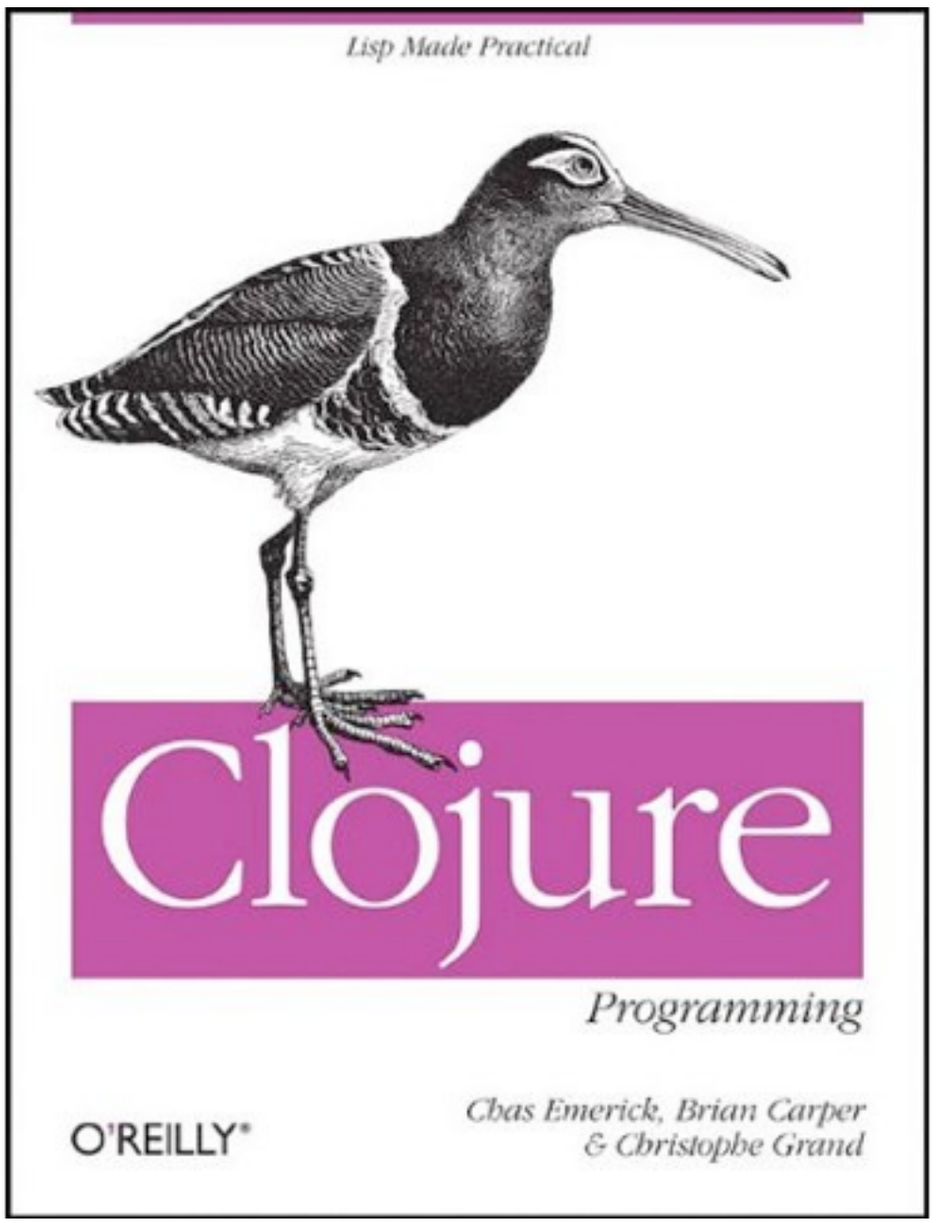


Snowtide

@cemerick

Clojure full-time since 2008

Contributor to Clojure language & libraries



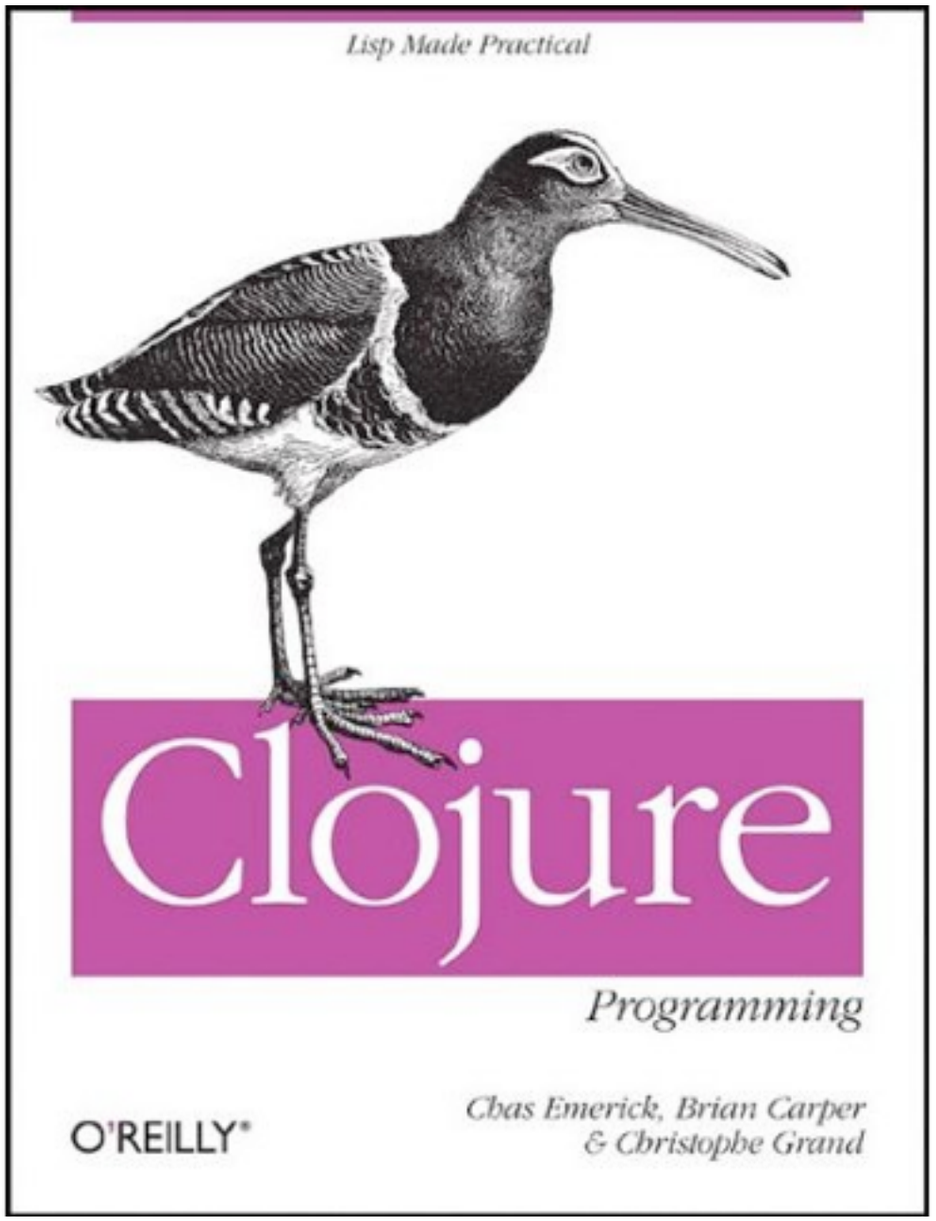
Snowtide

@cemerick

Clojure full-time since 2008

Contributor to Clojure language & libraries

Mostly λazy



Snowtide

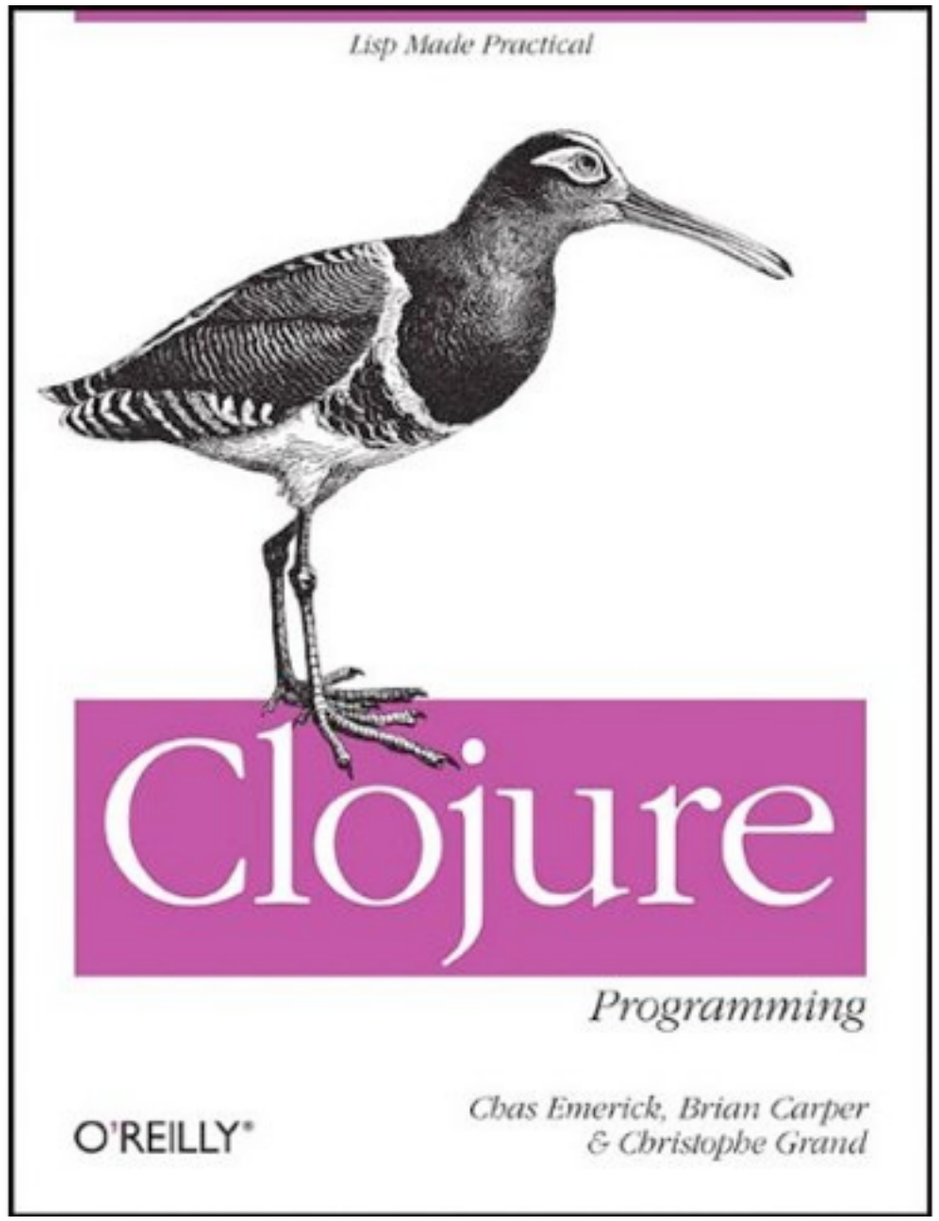
@cemerick

Clojure full-time since 2008

Contributor to Clojure language & libraries

Mostly λazy

Clojure Atlas



Snowtide

@cemerick

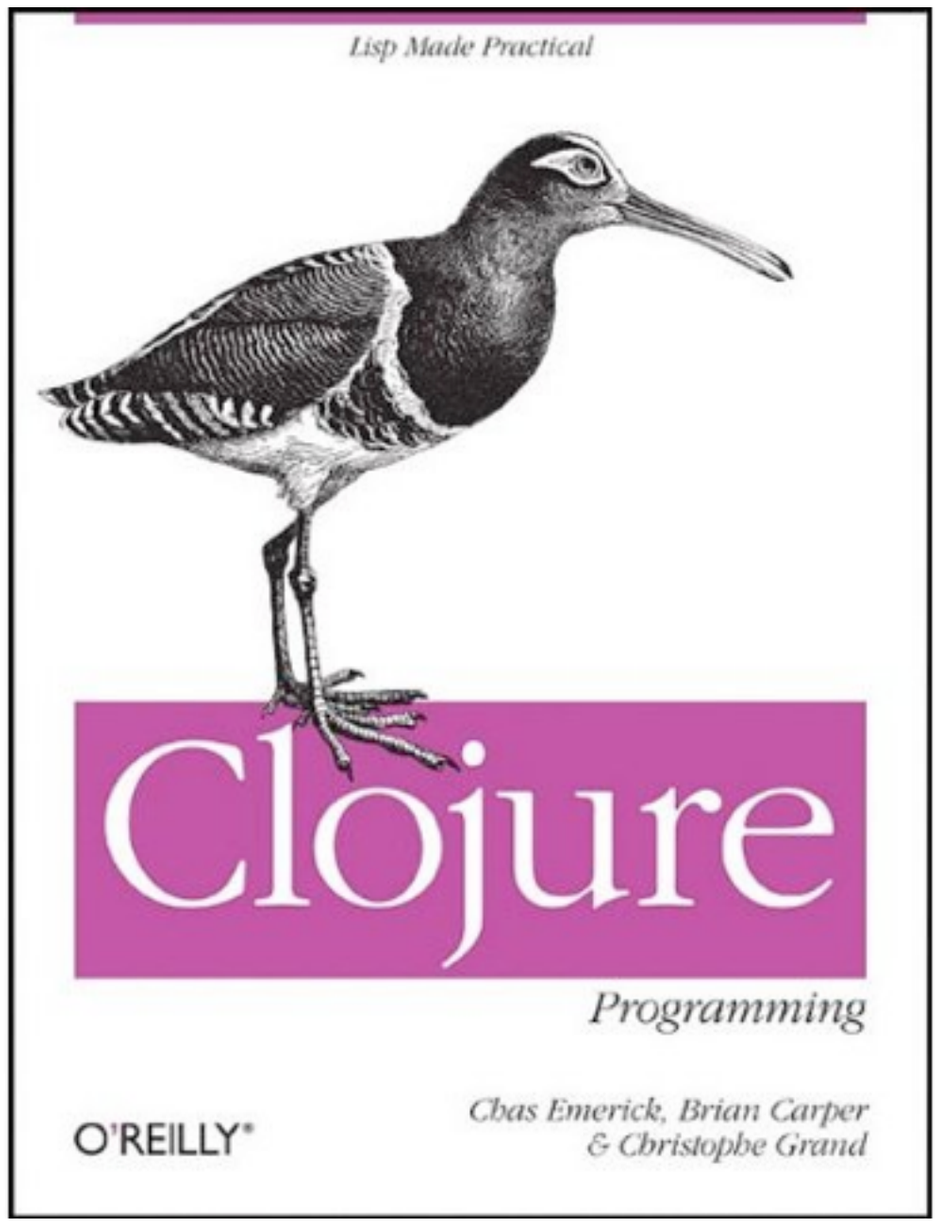
Clojure full-time since 2008

Contributor to Clojure language & libraries

Mostly λazy

Clojure Atlas

O'Reilly author



Snowtide

Why Bayesian networks?

Why Bayesian networks?

- Lots of modeling options

Why Bayesian networks?

- Lots of modeling options
  - Hand-curated, developed via Monte Carlo methods, or both (or other)

Why Bayesian networks?

- Lots of modeling options
  - Hand-curated, developed via Monte Carlo methods, or both (or other)
  - *No black boxes*

Why Bayesian networks?

- Lots of modeling options
  - Hand-curated, developed via Monte Carlo methods, or both (or other)
- *No black boxes*
  - Can determine root causes for individual inferences (*why do you think X?*)

Why Bayesian networks?

- Lots of modeling options
  - Hand-curated, developed via Monte Carlo methods, or both (or other)
- *No black boxes*
  - Can determine root causes for individual inferences (*why do you think X?*)
  - Can integrate with other systems at any level

Why Bayesian networks?

- Lots of modeling options
  - Hand-curated, developed via Monte Carlo methods, or both (or other)
  - *No black boxes*
    - Can determine root causes for individual inferences (*why do you think X?*)
    - Can integrate with other systems at any level
- Optimizable

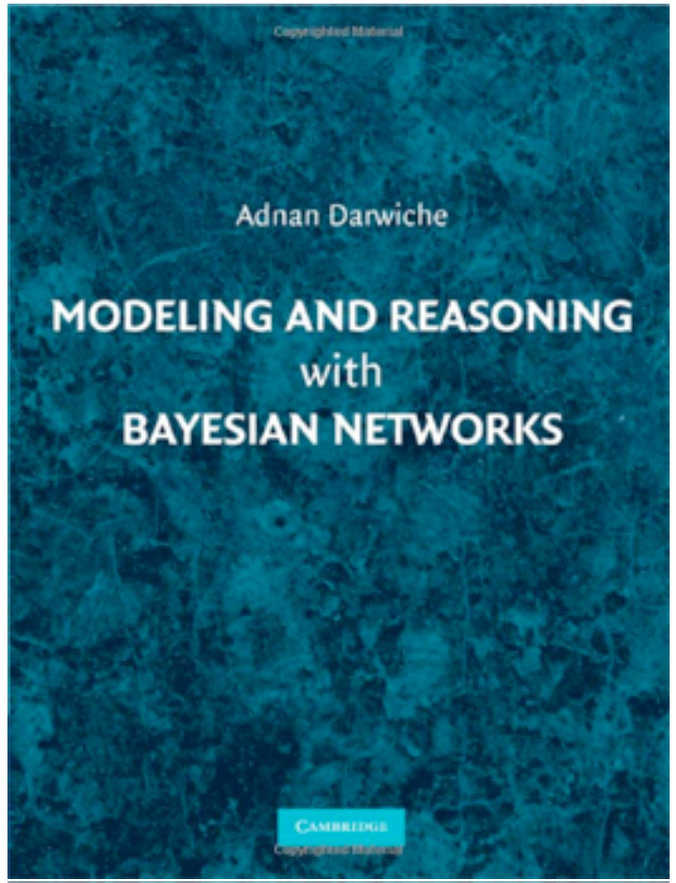
I'm a practitioner with problems to solve

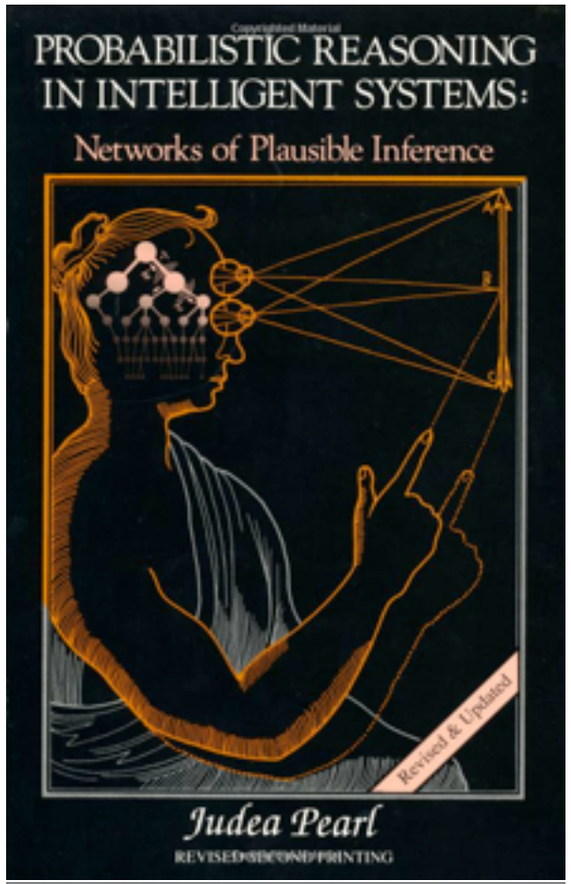
I'm a math idiot

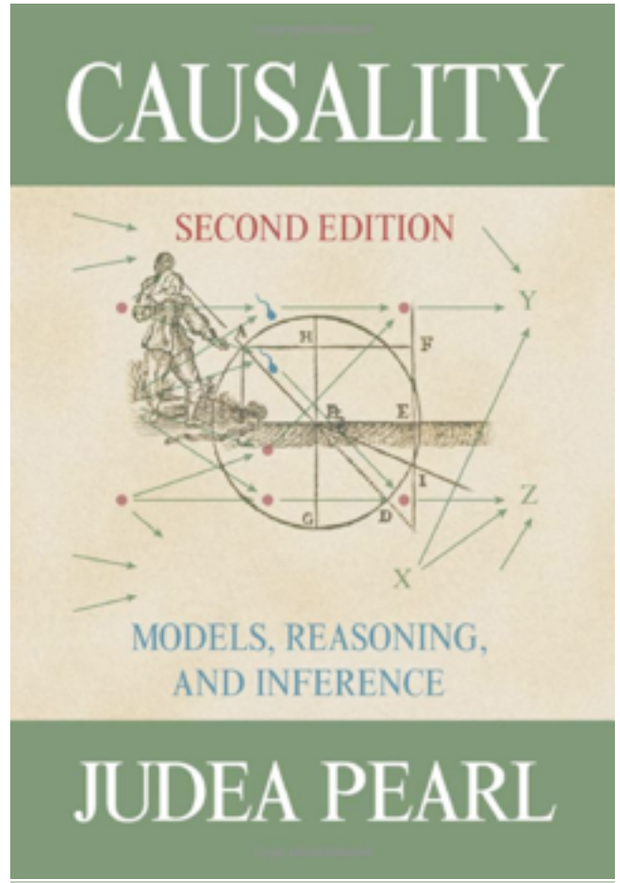
(shhhhh!)

Snowtide

Book porn


Copyrighted Material
Adnan Darwiche
MODELING AND REASONING
with
BAYESIAN NETWORKS
CAMBRIDGE
Copyrighted Material


Copyrighted Material
PROBABILISTIC REASONING
IN INTELLIGENT SYSTEMS:
Networks of Plausible Inference
Revised & Updated
Judea Pearl
REVISED SECOND PRINTING


CAUSALITY
SECOND EDITION
MODELS, REASONING,
AND INFERENCE
JUDEA PEARL

Snowtide

Domains & Applications

Domains & Applications

Building incomplete representations of the world, and making inferences and choices based on incomplete, erroneous, and noisy data.

Domains & Applications

Building incomplete representations of the world, and making inferences and choices based on incomplete, erroneous, and noisy data.

- Process control

Domains & Applications

Building incomplete representations of the world, and making inferences and choices based on incomplete, erroneous, and noisy data.

- Process control
- Decision systems

Domains & Applications

Building incomplete representations of the world, and making inferences and choices based on incomplete, erroneous, and noisy data.

- Process control
- Decision systems
- Prediction

Domains & Applications

Building incomplete representations of the world, and making inferences and choices based on incomplete, erroneous, and noisy data.

- Process control
- Decision systems
- Prediction
- Classification

Document analysis

	February 28, 2006	
	Shares	Value
Natural Gas Gathering/Processing – 4.4%(1)		
Energy Transfer Partners, L.P.	95,200	$ 3,400,544
Regency Energy Partners, L.P.	82,500	1,654,950
		5,055,494
Natural Gas/Natural Gas Liquid Pipelines – 7.5%(1)		
Enterprise Products Partners, L.P.	217,210	5,273,859
Northern Border Partners, L.P.	71,000	3,415,100
		8,688,959
Propane Distribution – 0.4%(1)		
Inergy, L.P.	17,365	474,759
Total Master Limited Partnerships and Related Companies (Cost $50,648,748)		50,860,146
	Principal Amount	
Corporate Bonds – 10.2%(1)		
Crude/Refined Products Pipeline – 6.6%(1)		
SemGroup, L.P., 8.75%, 11/15/2015(4)	$ 7,300,000	7,555,500
Electric Generation/Services – 1.8%(1)		
NRG Energy, Inc., 7.25%, 2/1/2014	1,000,000	1,025,000
NRG Energy, Inc., 7.375%, 2/1/2016	1,000,000	1,030,000

Document analysis

PARTY IN INTEREST -
IDENTITY - ASSET ALLOCATION FD
DESCRIPTION - RIC
COST - 0
CURRENT VALUE - 16332

PARTY IN INTEREST -
IDENTITY - BLUE CHIP GROWTH FD
DESCRIPTION - RIC
COST - 0
CURRENT VALUE - 60760

PARTY IN INTEREST -
IDENTITY - ARIEL APPRECIATION
DESCRIPTION - REG INVEST CO (RIC)
COST - 0
CURRENT VALUE - 19500

PARTY IN INTEREST -
IDENTITY - ARIEL FUND
DESCRIPTION - RIC
COST - 0
CURRENT VALUE - 9969

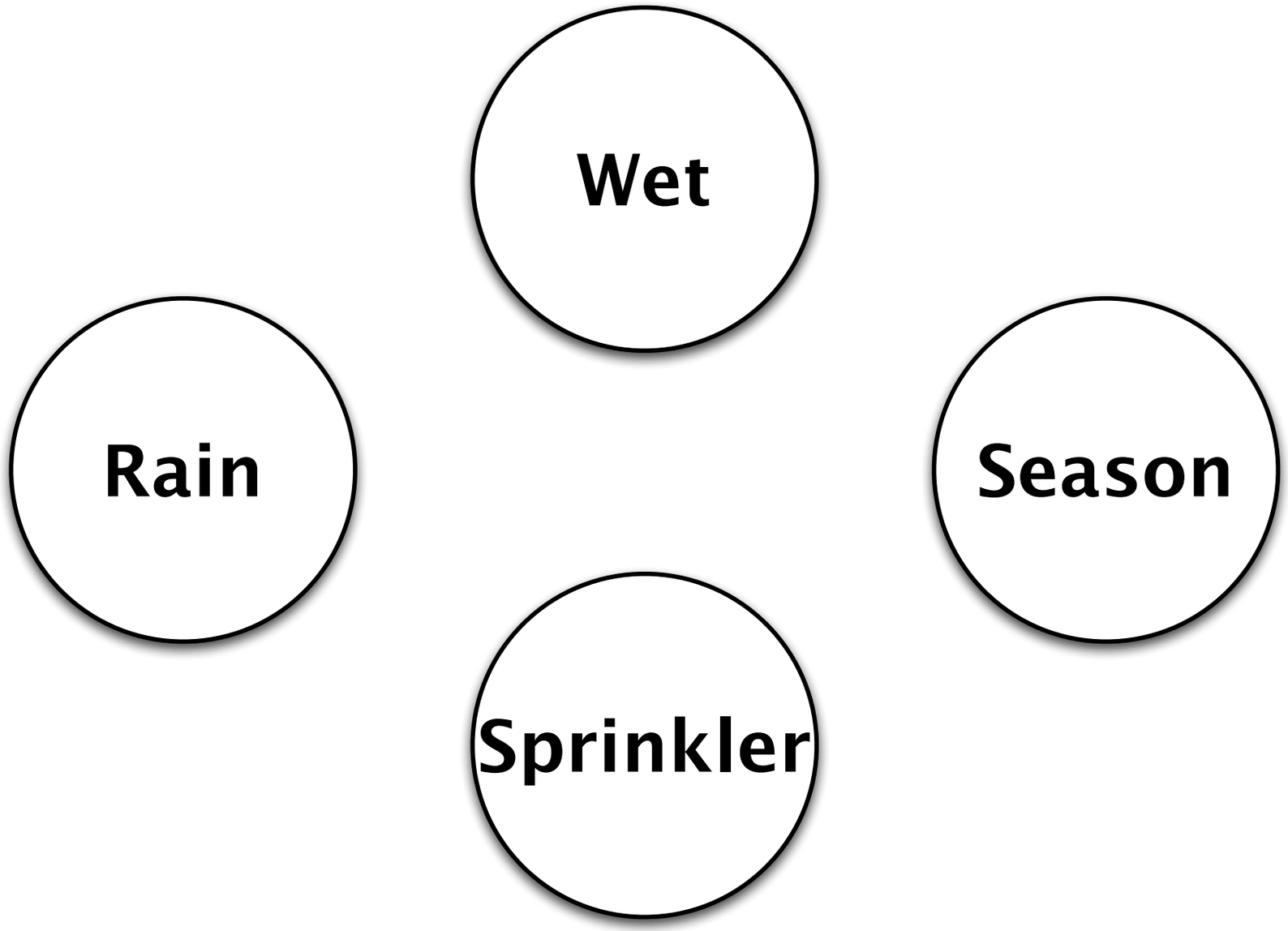
Document analysis

STRATEGIC ALLOCATION: CONSERVATIVE - SCHEDULE OF INVESTMENTS
FEBRUARY 28, 2006 (UNAUDITED)

SHARES		VALUE
BIOTECHNOLOGY - 0.4%		
6,019	Alkermes Inc.(1)	152,943
25,469	Amgen Inc.(1)	1,922,655
3,835	Applera Corporation-Applied Biosystems Group	108,415
1,900	Genentech, Inc.(1)	162,811
6,900	Genzyme Corp.(1)	478,446
110	Gilead Sciences, Inc.(1)	6,850
		2,832,120
BUILDING PRODUCTS - 0.2%		
5,700	Daikin Industries Ltd. ORD	191,100
20,700	Masco Corp.	645,633
4,948	USG Corp.(1)	418,007
		1,254,740

Features

(random variables)



Snowtide

Appeal to logic

Wet?	Raining?	Sprinkler?	Spring?
T	T	T	T
T	T	T	F
…			
F	F	F	F

Appeal to logic

Wet?	Raining?	Sprinkler?	Spring?	Spring + Rain?
T	T	T	T	T
T	T	T	F	F
…				
F	F	F	F	F

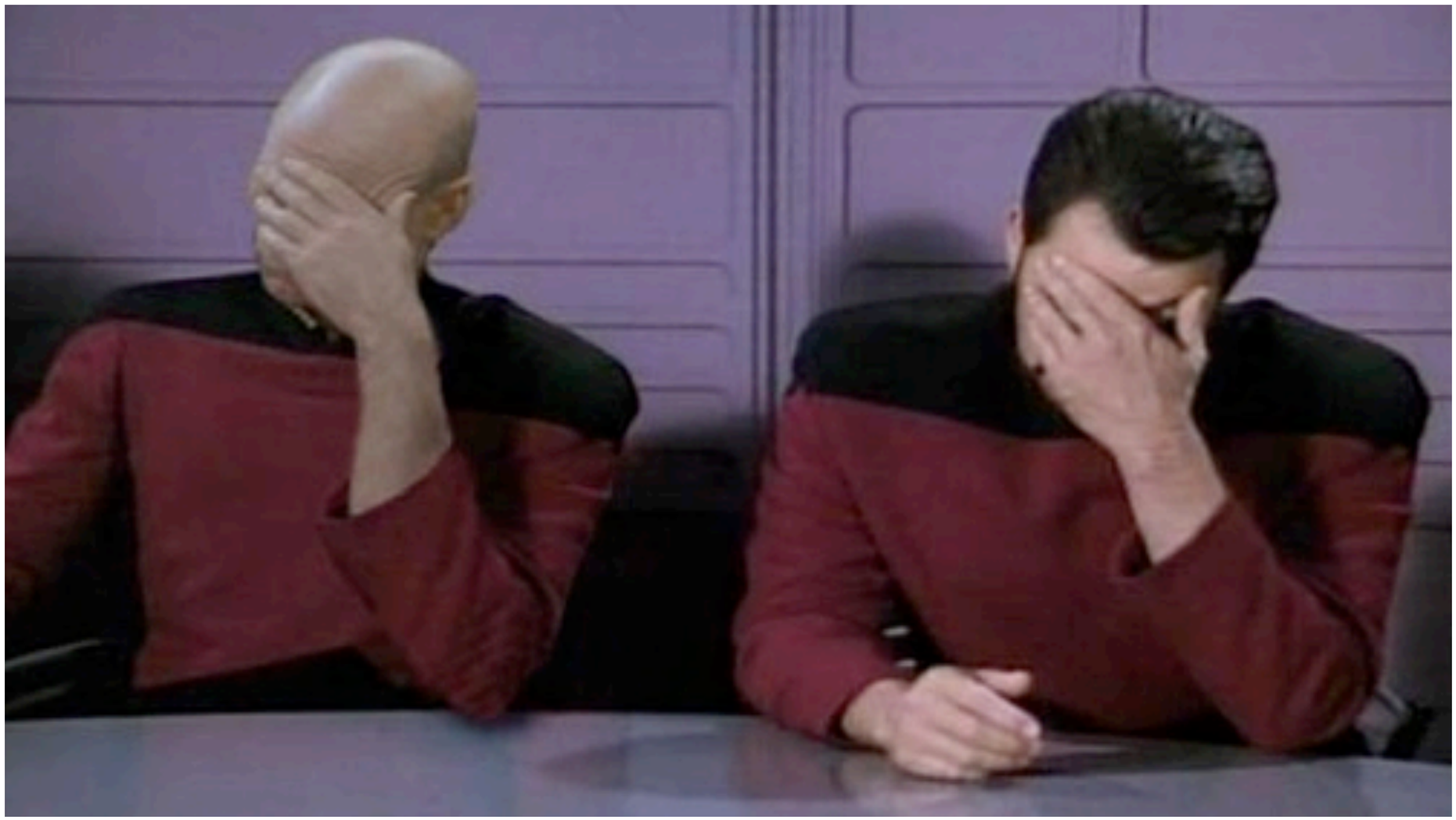
Sometimes logic isn't enough

Joint probability distribution

world	Wet?	Raining?	Sprinkler?	Spring?	P(ω)
ω_1	T	T	T	T	0.005
ω_2	T	T	T	F	0.01
		...			
ω_n	F	F	F	F	0.31

Probability in 10 minutes

panic



(Bayesian) probability in 1 minute

Bayesian probability

$$P(wet|spring) = \frac{P(spring|wet)P(wet)}{P(spring)}$$

JPDs don't scale

JPDs don't scale

- m^n worlds

JPDs don't scale

- m^n worlds
- {spring,wet,sprinkler,rain} = 2^4 = 16

JPDs don't scale

- m^n worlds
- {spring,wet,sprinkler,rain} = 2^4 = 16
- e.g. 20 variables x 3 states:

JPDs don't scale

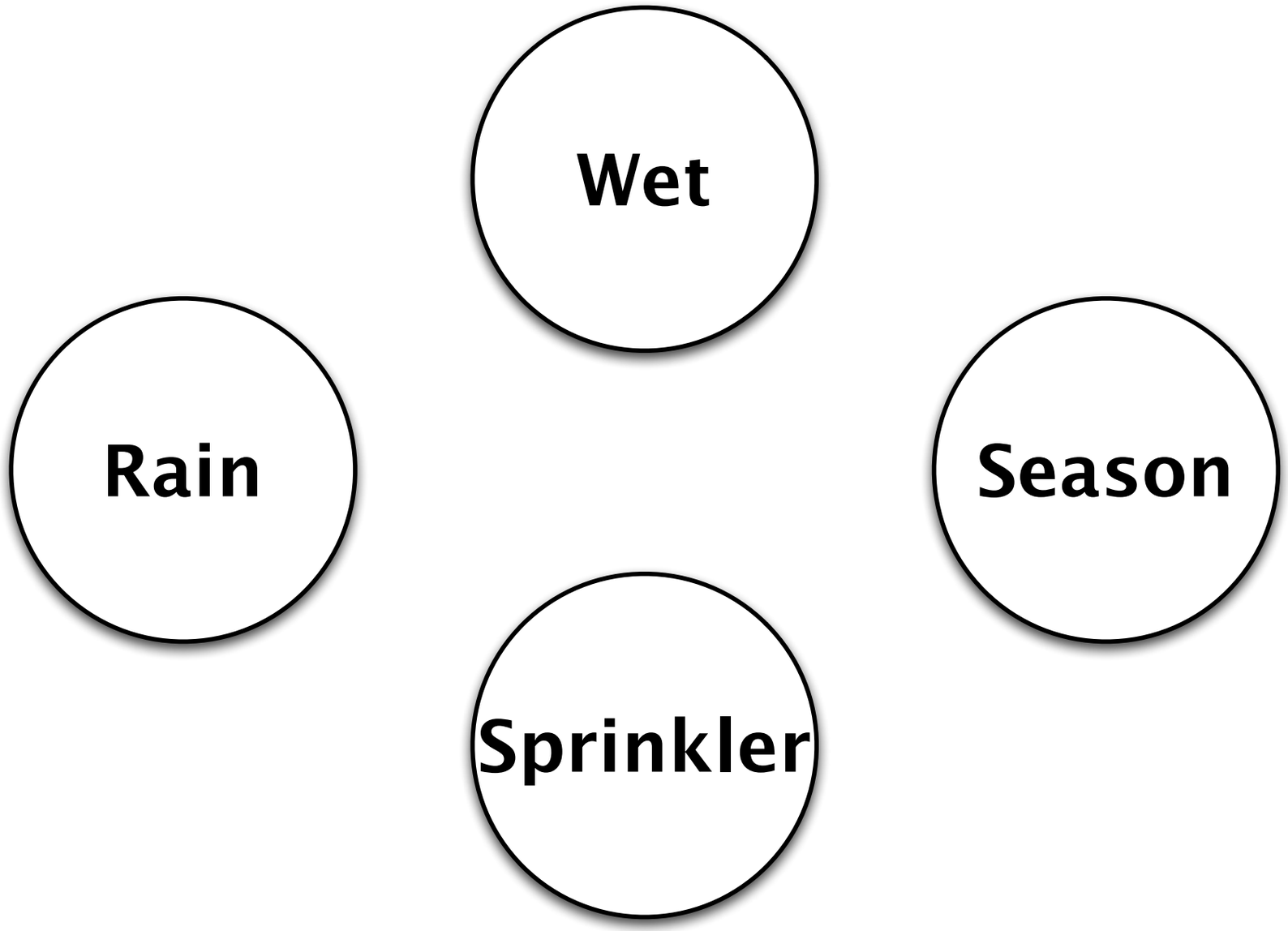
- m^n worlds
- {spring,wet,sprinkler,rain} = 2^4 = 16
- e.g. 20 variables x 3 states:
  - 3 ^ 20 = 3.486784401E9

JPDs don't scale

- m^n worlds
- {spring,wet,sprinkler,rain} = 2^4 = 16
- e.g. 20 variables x 3 states:
  - 3 ^ 20 = 3.486784401E9
- Need a better representation

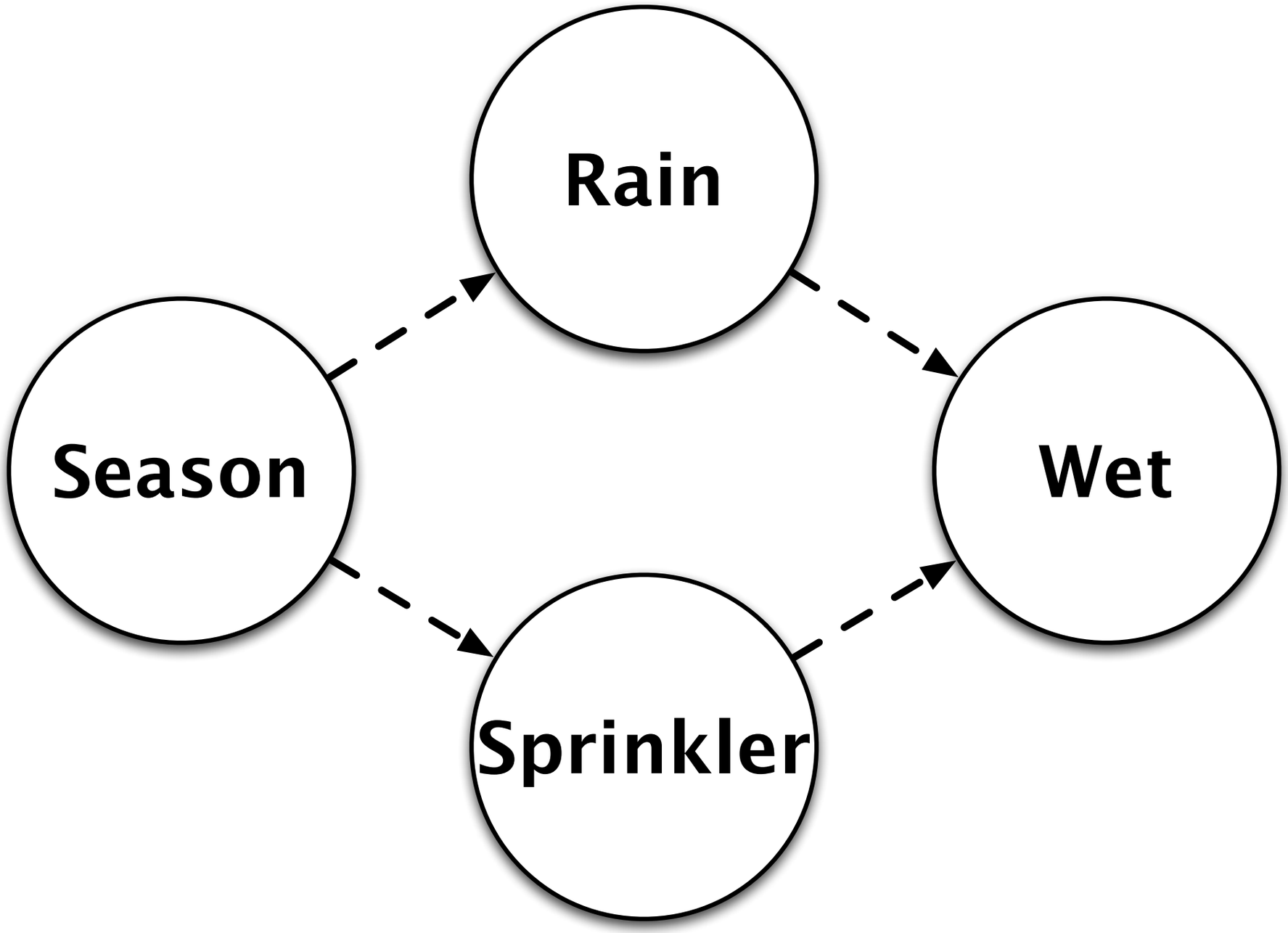
Features

(random variables)



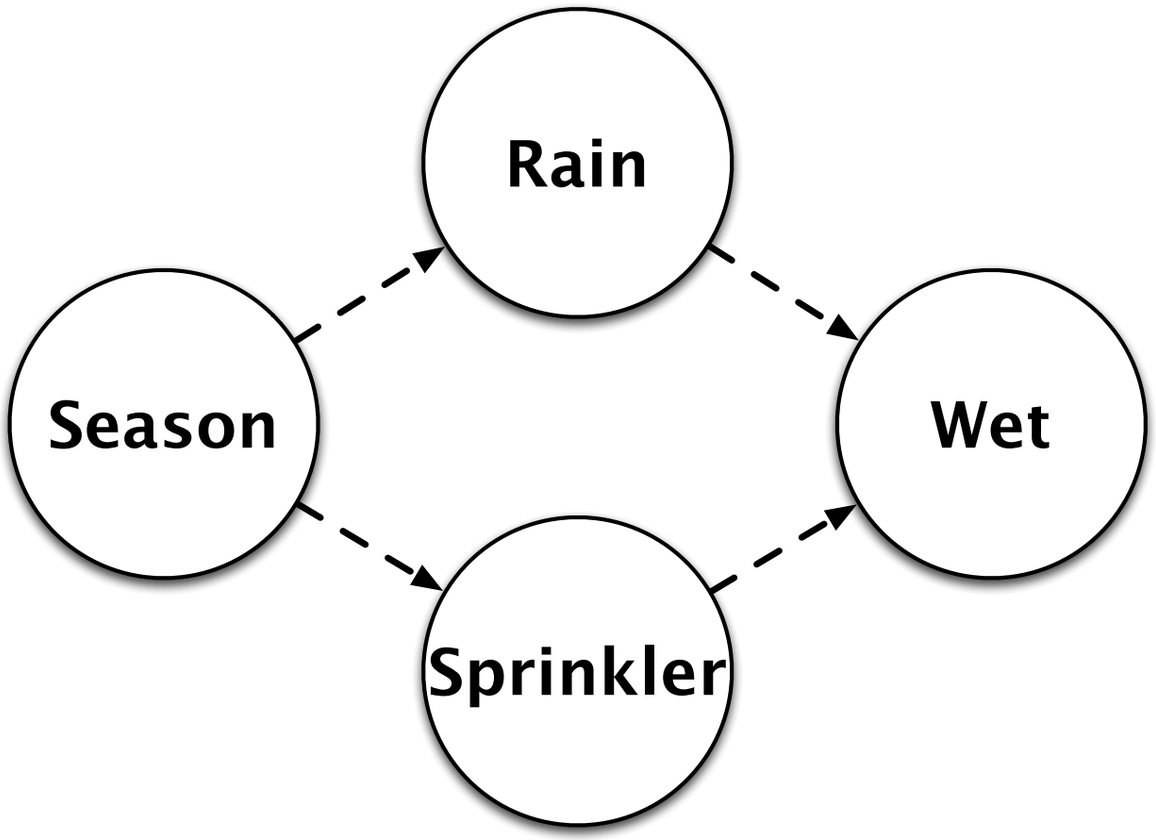
Snowtide

Bayesian Network



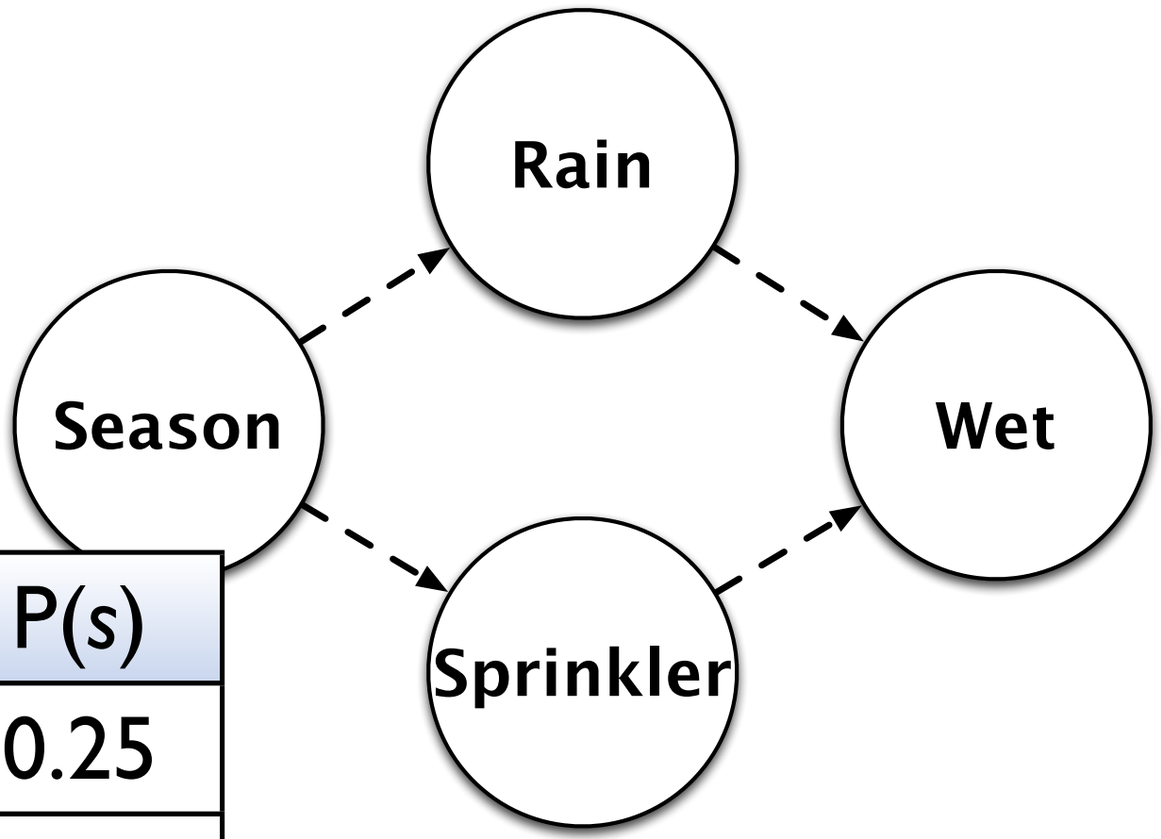
Snowtide

Network effects



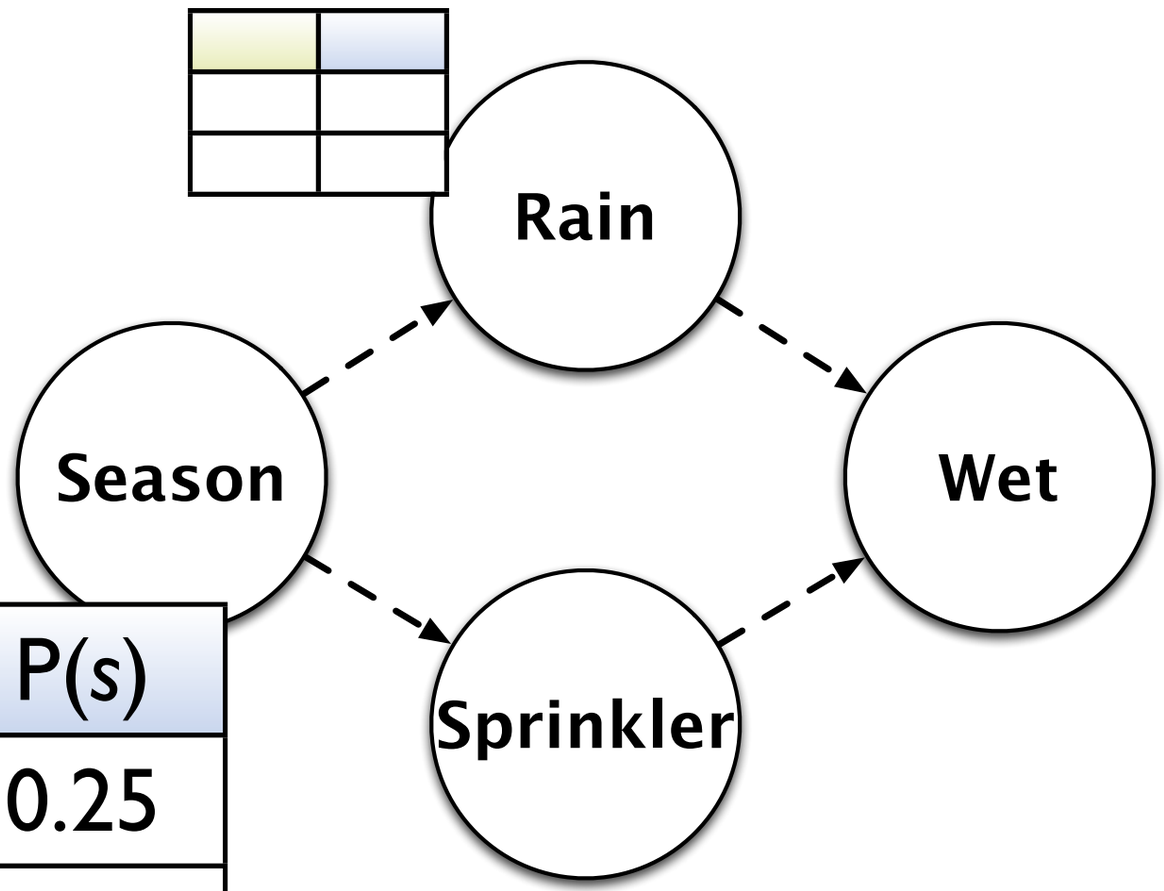
Snowtide

Network effects



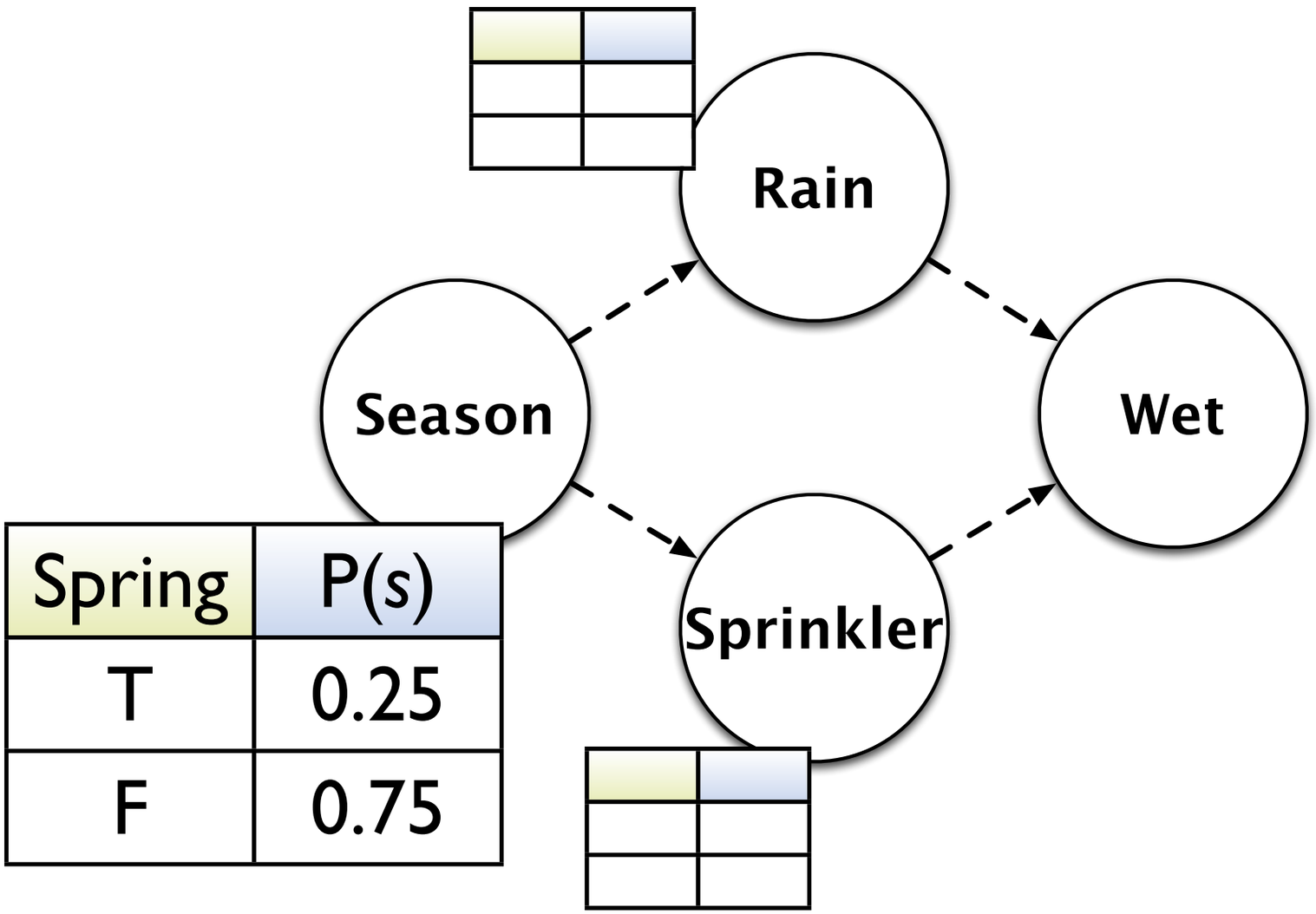
Spring	P(*s*)
T	0.25
F	0.75

Network effects



Spring	P(*s*)
T	0.25
F	0.75

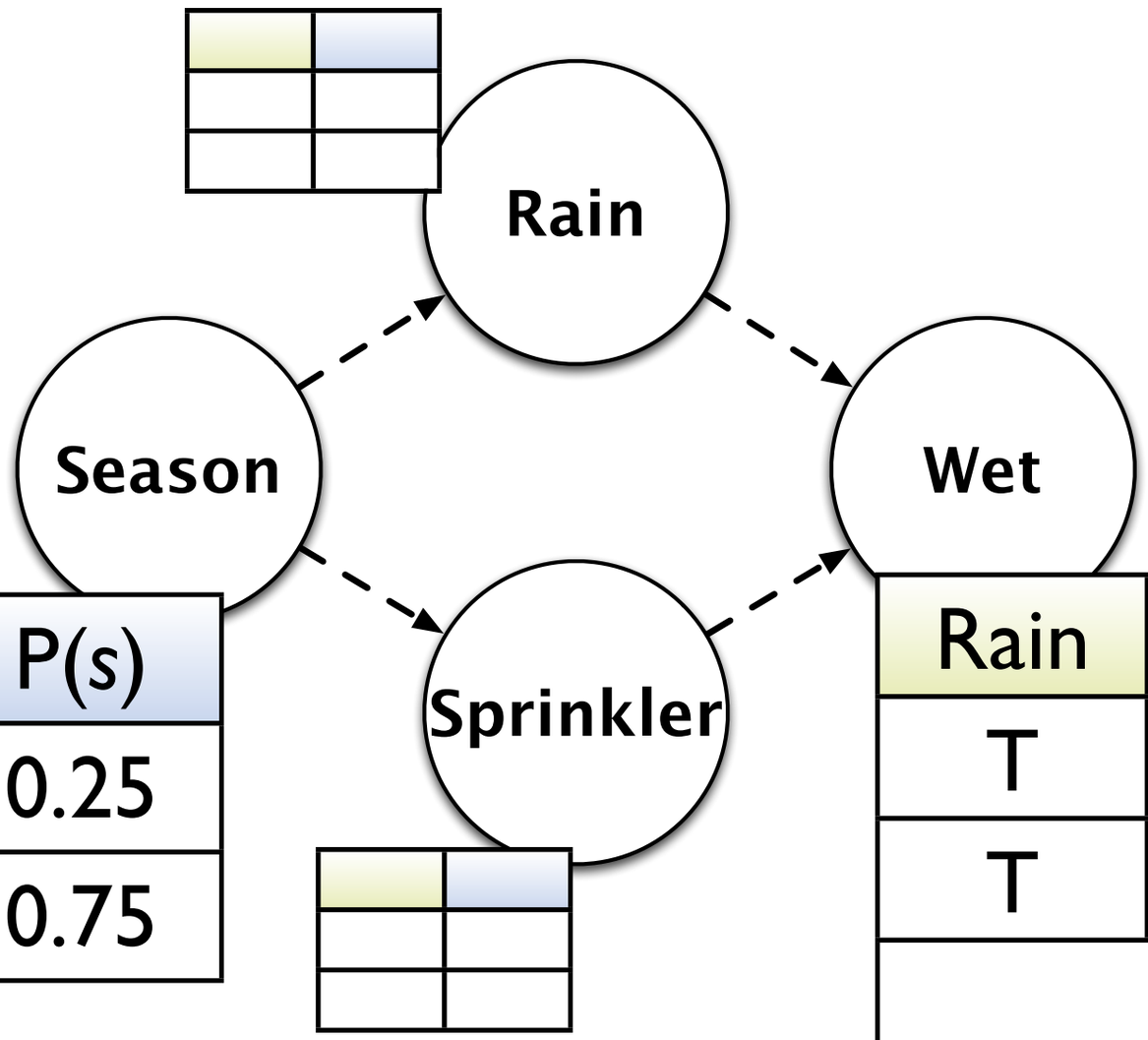
Network effects



Spring	P(*s*)
T	0.25
F	0.75

Snowtide

Network effects

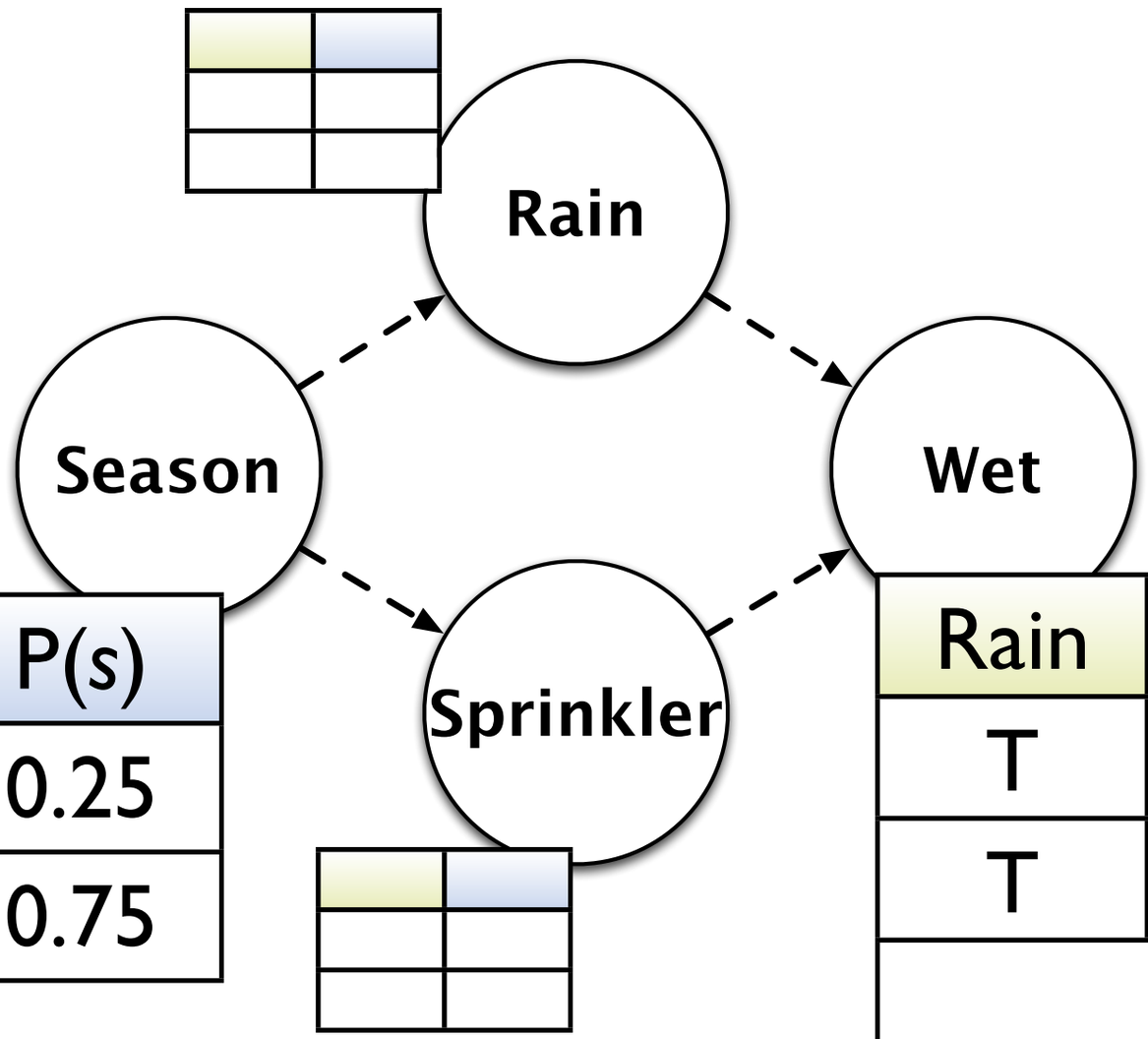


Spring	P(*s*)
T	0.25
F	0.75

Rain	Sprinkler	Wet	P(*w*|*r*∧*s*)
T	T	T	0.07
T	F	T	0.18
…			
F	F	F	0.31

Snowtide

Network effects



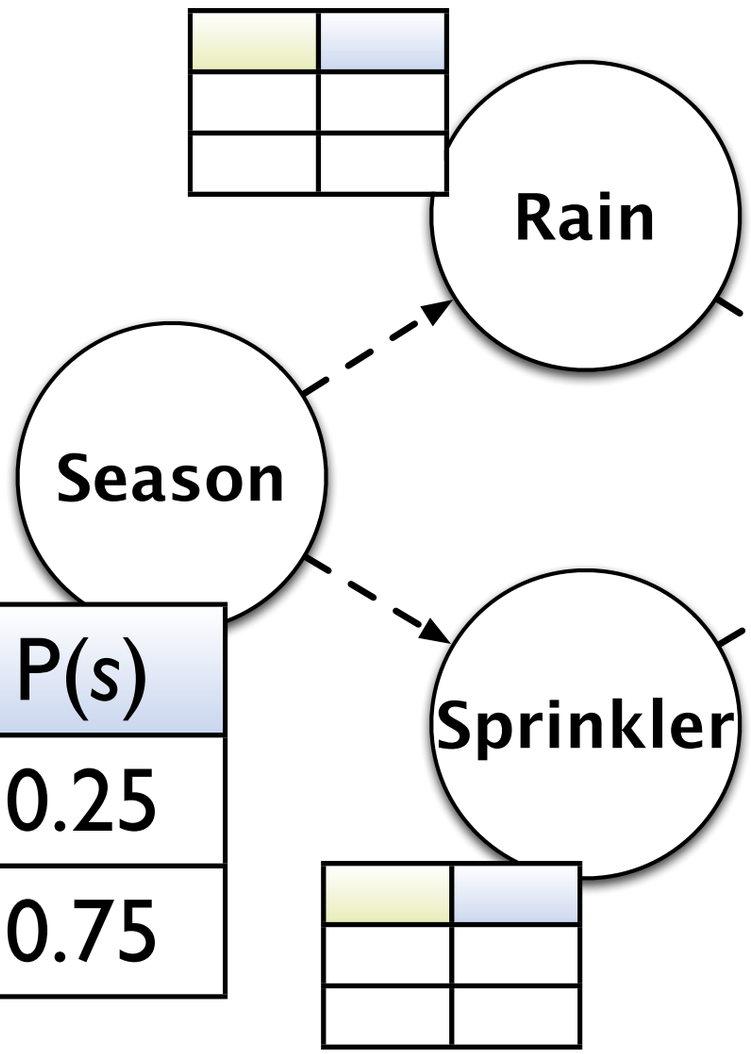
Spring	P(*s*)
T	0.25
F	0.75

Rain	Sprinkler	Wet	P(*w*|*r*∧*s*)
T	T	T	0.07
T	F	T	0.18
…			
F	F	F	0.31

$O(n \log_m(p))$!

Snowtide

Network effects



Spring	P(*s*)
T	0.25
F	0.75

Rain	Sprinkler	Wet	P(*w*|*r*∧*s*)
T	T	T	0.07
T	F	T	0.18
…			
F	F	F	0.31

$O(n \log_m(p))!$

(n, m, p left as an exercise) ;-)

Snowtide

Where's the f#&@*$ Clojure?

Raposo

Raposo

- Clojure library for Bayesian inference and modeling

Raposo

- Clojure library for Bayesian inference and modeling
- Reimplementation / extraction from existing application

Raposo

- Clojure library for Bayesian inference and modeling
- Reimplementation / extraction from existing application
- Clojure-idiomatic treatment of data

Raposo

- Clojure library for Bayesian inference and modeling
- Reimplementation / extraction from existing application
- Clojure-idiomatic treatment of data
- Learning vehicle

Data

```
[#{:rain :wet}
 #{:rain}
 #{:sprinkler :wet}
 #{:rain :wet}
 #{:sprinkler}
 #{:rain :wet}]

[{:wet true :rain 1.8}
 {:wet false :rain 0.1}
 {:sprinkler 2.0 :wet true}
 {:sprinkler 0.1 :rain 0.2 :wet false}
 {:sprinkler 0.5 :rain 1.1 :wet true}]
```

An abstraction over data

An abstraction over data

```
(defprotocol Observed
  (features [this])
  (value [this feature-name]))

(extend-protocol Observed
  clojure.lang.IPersistentSet
  (features [this] this)
  (value [this feature-name]
    (.contains this feature-name))

  clojure.lang.IPersistentMap
  (features [this] (keys this))
  (value [this feature-name]
    (.valAt this feature-name)))
```

An abstraction over data

```
(defprotocol Observed
  (features [this])
  (value [this feature-name]))

(extend-protocol Observed
  clojure.lang.IPersistentSet
  (features [this] this)
  (value [this feature-name]
    (.contains this feature-name))

  clojure.lang.IPersistentMap
  (features [this] (keys this))
  (value [this feature-name]
    (.valAt this feature-name)))
```

(Allows us to bridge existing feature-ful data types to Raposo without copying.)

Modeling the world

```
(def model (create-model {:rain #{:wet}
                          :spring #{:rain :sprinkler}
                          :sprinkler #{:wet}
                          :wet}))

(def populated-model (into model [#{:rain :wet}
                                  #{:rain :spring}
                                  #{:sprinkler :wet}
                                  #{:rain :wet}
                                  #{:sprinkler}
                                  #{:rain :wet}]))
```

Modeling the world (MCMC)

```
(def model (create-model [{:wet true :rain 1.8 :spring true}
                          {:wet false :rain 0.1}
                          {:sprinkler 2.0 :wet true :spring false}
                          {:sprinkler 0.1 :rain 0.2 :wet false}
                          {:sprinkler 0.5 :rain 1.1 :wet true}
                          ...]))
```

Query

Query

```
=> (probability-of :wet
     model {:rain 0.9 :sprinkler 0.6})
0.8483736484567366
```

Query

```
=> (probability-of :wet
     model {:rain 0.9 :sprinkler 0.6})
0.8483736484567366
```

```
=> (probability-of :wet
     model {:sprinkler false})
0.48372656372001
```

Query

```
=> (probability-of :wet
      model {:rain 0.9 :sprinkler 0.6})
0.8483736484567366
```

```
=> (probability-of :wet
      model {:sprinkler false})
0.48372656372001
```

```
=> (probability-of :rain
      model {:wet true :sprinkler 0.0})
0.912384736226474
```

Raposo TODO

Raposo TODO

- Release!

Raposo TODO

- Release!
- Expose network learning / generation

Raposo TODO

- Release!
- Expose network learning / generation
- Optimization

Raposo TODO

- Release!
- Expose network learning / generation
- Optimization
- Elision of independent random variables

Raposo TODO

- Release!
- Expose network learning / generation
- Optimization
  - Elision of independent random variables
  - Compilation

Raposo TODO

- Release!
- Expose network learning / generation
- Optimization
  - Elision of independent random variables
  - Compilation
    - Bayesian network => Clojure

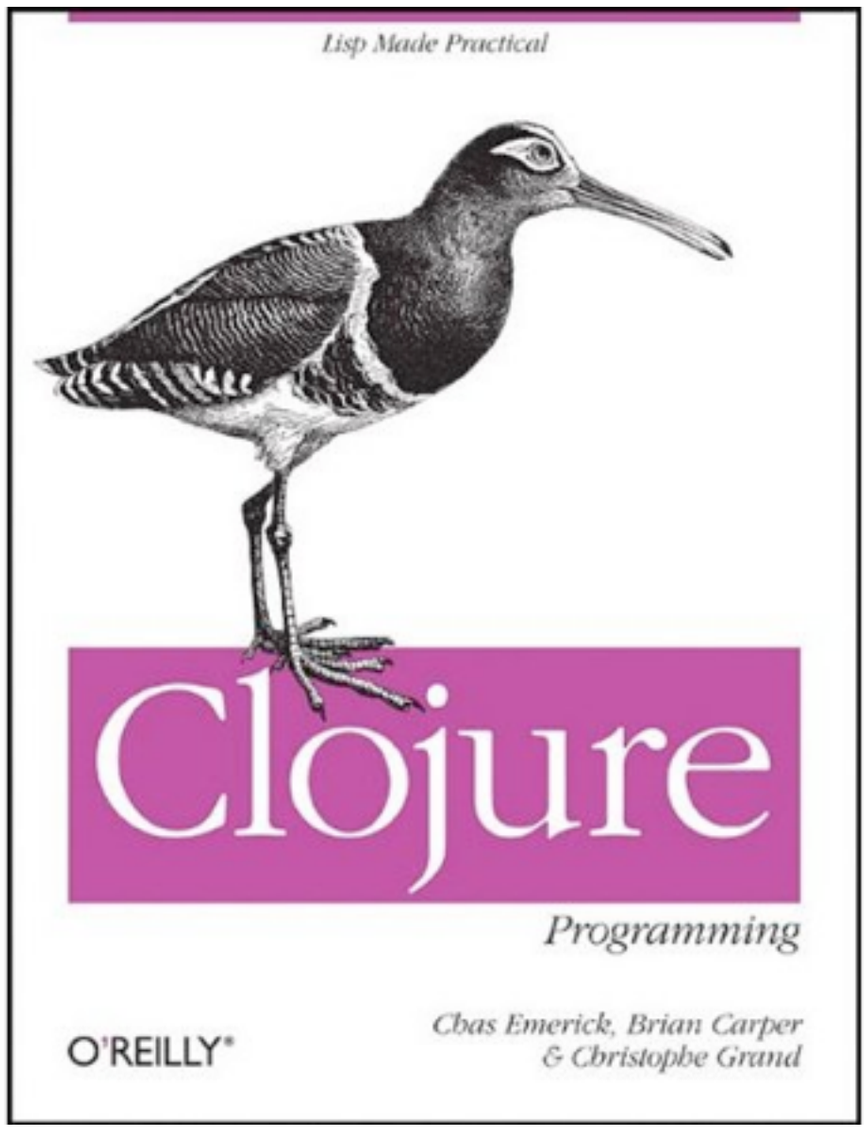
Raposo TODO

- Release!
- Expose network learning / generation
- Optimization
  - Elision of independent random variables
  - Compilation
    - Bayesian network => Clojure
- Integrate with `core.logic`

Thank you!

Chas Emerick
@cemerick
cemerick.com
snowtide.com

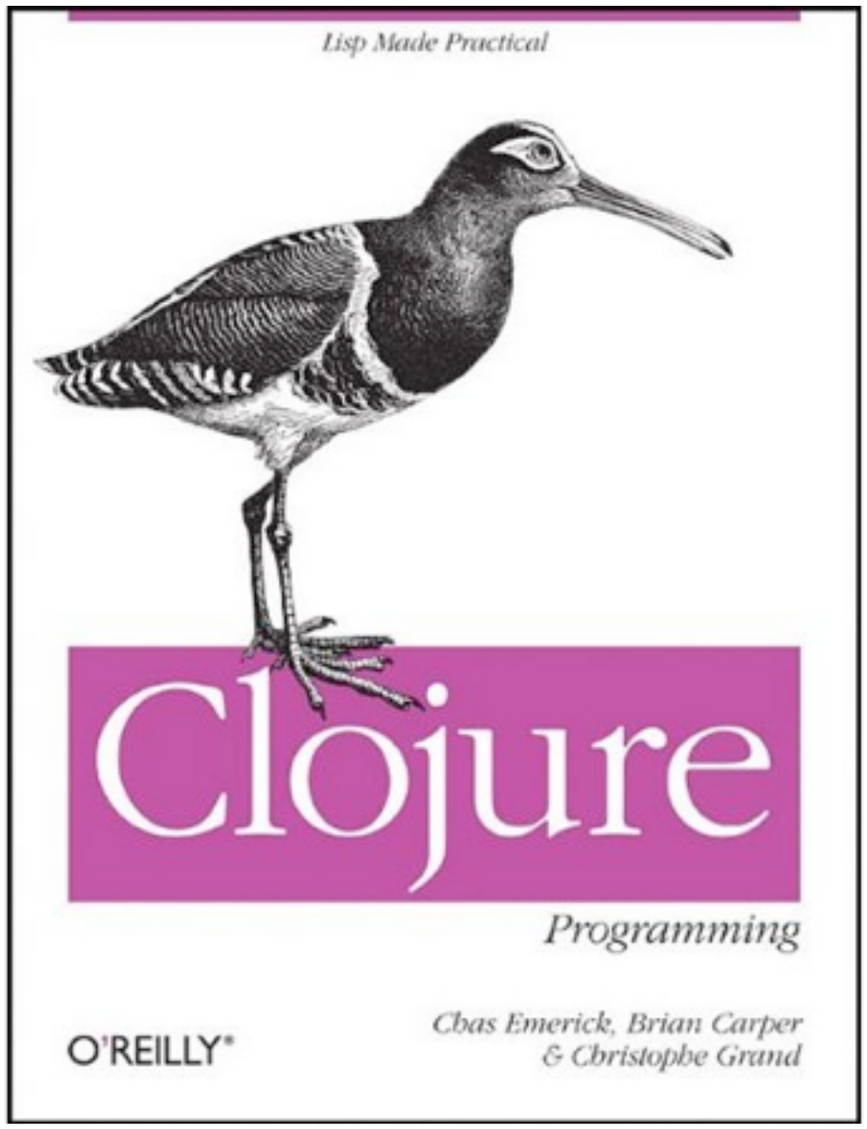
clojureatlas.com
clojurebook.com
mostlylazy.com



Snowtide

Thank you!

Raposo (coming soon):



Chas Emerick
@cemerick
cemerick.com
snowtide.com

clojureatlas.com
clojurebook.com
mostlylazy.com

Snowtide

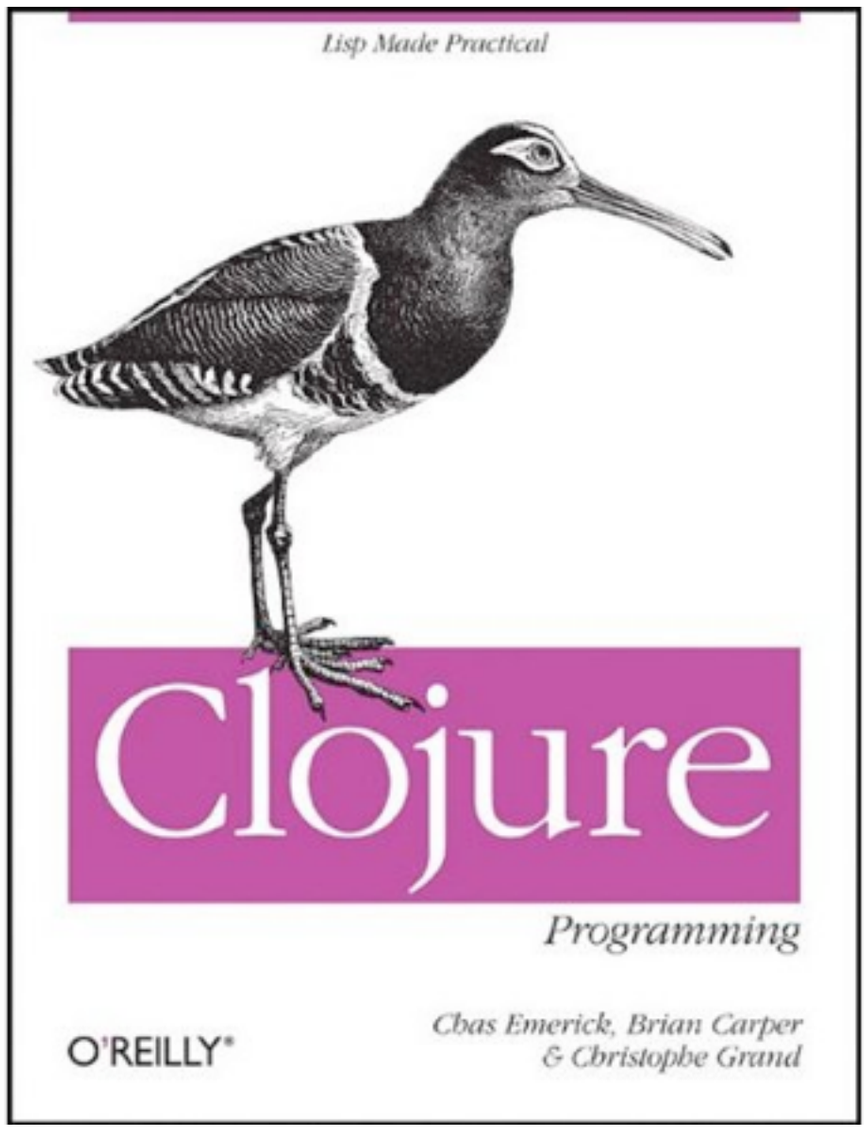
Thank you!

Raposo (coming soon):

http://github.com/cemerick/raposo

Chas Emerick
@cemerick
cemerick.com
snowtide.com

clojureatlas.com
clojurebook.com
mostlylazy.com



Snowtide