Soft Concepts vs Hard Facts: software models in the social sciences

Cezar Ionescu
cezar.ionescu@conted.ox.ac.uk

OU Alumni Weekend, 15th September 2018
`https://github.com/ionescu/alumni_weekend_2018`

Programming languages

Idris, Agda

ML, Haskell

Python, R

C++, Java

Algol, Pascal, C

assembly language

machine code

Abstraction

Repeating patterns in the lower levels were named, abstracted out, and made into building blocks of the higher levels.

Abstraction

Repeating patterns in the lower levels were named, abstracted out, and made into building blocks of the higher levels.

Patterns of *flow of control*:

Abstraction

Repeating patterns in the lower levels were named, abstracted out, and made into building blocks of the higher levels.

Patterns of *flow of control*:

- subroutines with goto
- procedure calls
- objects

Patterns of *data*:

Abstraction

Repeating patterns in the lower levels were named, abstracted out, and made into building blocks of the higher levels.

Patterns of *flow of control*:

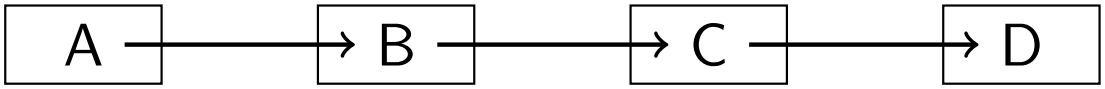
- subroutines with goto
- procedure calls
- objects

Patterns of *data*:

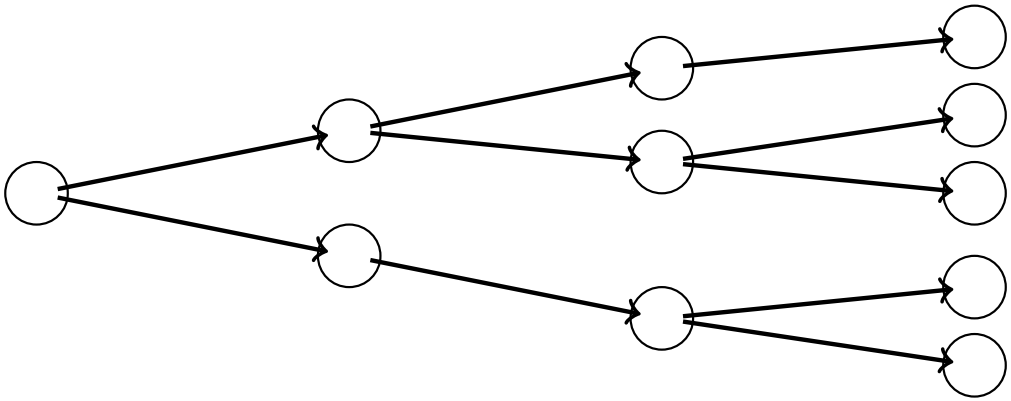
- arrays
- lists
- trees
- graphs

Example: data structures and traversal

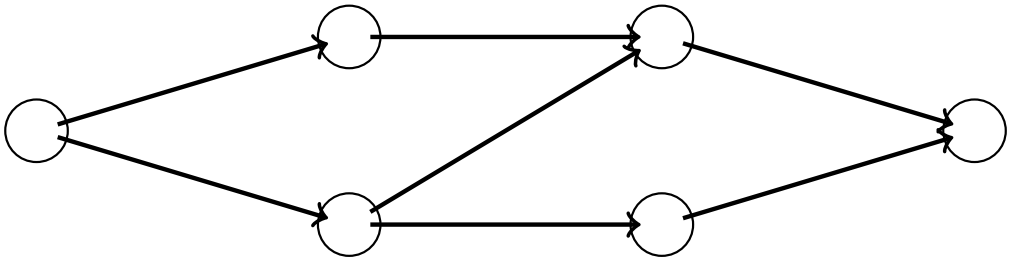
A list:



A tree:



A graph:



Frameworks

Patterns of *systems building*:

- standard libraries (C standard library, Java APIs, C++ STL, . . .)
- frameworks (Ruby on Rails, .Net Framework, TensorFlow, . . .)
- stacks (LAMP, Elastic Stack, . . .)

“Law” of conservation of errors

We have better programing languages, better tools, better training, better processes.

However, the systems we build seem to have (at least) as many bugs as those of the past.

“Law” of conservation of programming errors: program complexity increases faster than our ability to control complexity.

Ad-hoc abstractions

Systems are built on towers of abstractions; correctness must be ensured at **each** level.

Many abstractions had in the beginning an "ad-hoc" character

Example: object-oriented programming.

Program correctness

Edsger Wybe Dijkstra

Source: `https://en.wikipedia.org/wiki/File:Edsger_Wybe_Dijkstra.jpg`

Program testing can be used to show the presence of bugs, but never to show their absence!

Dijkstra (1970)

Alas, by 1975 Dijkstra [. . .] was no longer interested in programs that were too large to be proven mathematically correct. (This pretty much ruled out any program of more than a couple of pages).

Per Brinch Hansen (2004)

Mathematics for Computer Science

For safety-critical systems, we do require programs that are mathematically proven correct.

This has led to investigating less "ad-hoc" and more mathematical abstractions.

Category theory and *type theory* are the primary source of such abstractions. They deal with very abstract forms of putting together simple things to make complex things.

Example: type generic programming.

Programming languages based on these theories make it easier to ensure program correctness.

Specifications of "soft concepts"

However, in order to use these we need *formal specifications*.

We have informal requirements for systems such as loan approval assistants (fairness, non-discrimination, ...), self-driving cars (safety, respectful of other drivers, ...), autonomous weapons (ethical behaviour), and so on.

It is not obvious that such requirements can be formalised.

The Potsdam Institute for Climate Impact Research

The Potsdam Institute for Climate Impact Research

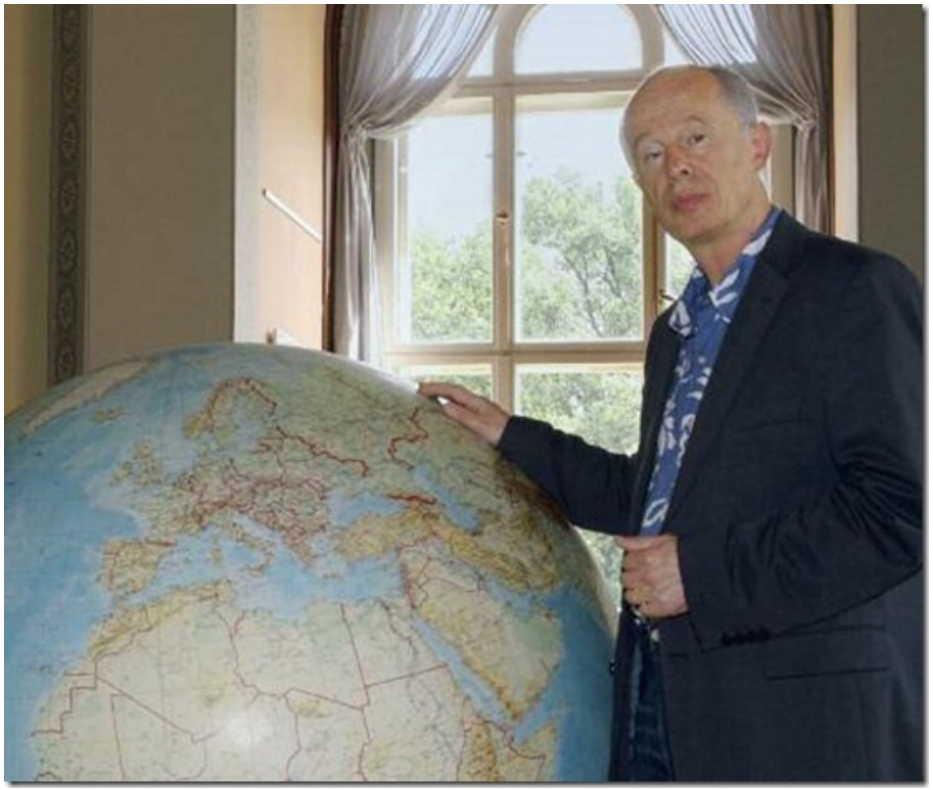
PIK addresses crucial scientific questions in the fields of global change, climate impacts and sustainable development.

Researchers from the natural and social sciences work together to generate interdisciplinary insights and to provide society with sound information for decision making.

The main methodologies are systems and scenarios analysis, modelling, computer simulation, and data integration.

PIK Mission, www.pik-potsdam.de, retrieved 2018-09-15

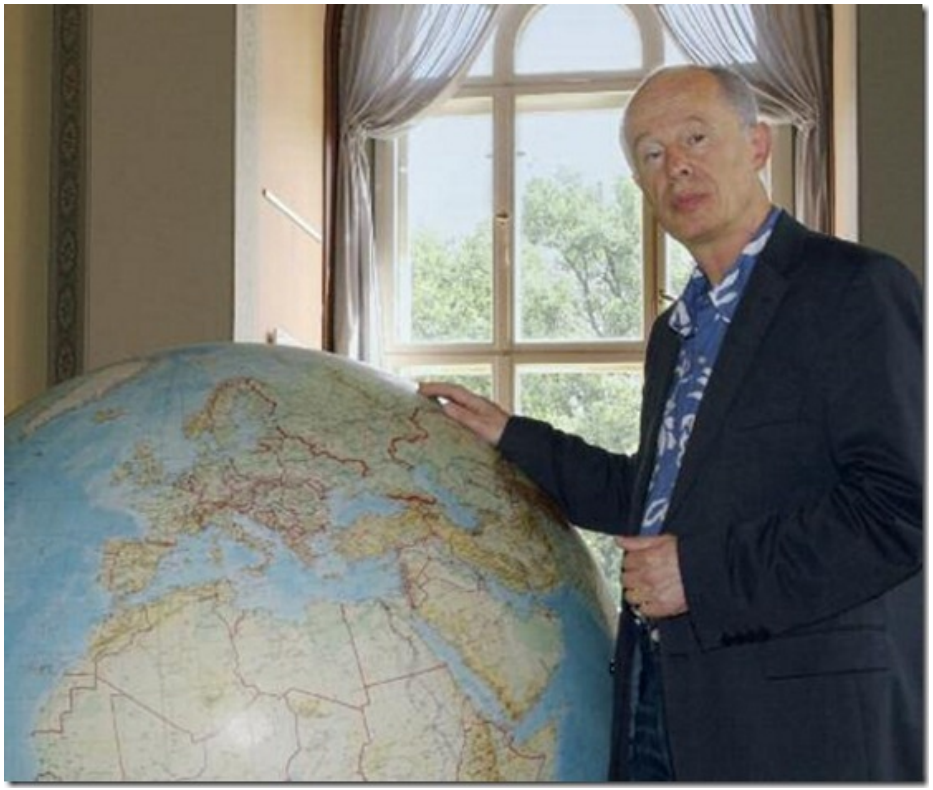
The Potsdam Institute for Climate Impact Research



"Die Rolle der Klimaforschung bleibt weiterhin, die Problemfakten auf den Tisch zu knallen und Optionen für geeignete Lösungswege zu identifizieren."

H.-J. Schellnhuber in *Frankfurter Allgemeine* from 2012-06-19

The Potsdam Institute for Climate Impact Research



"The role of the climate researcher continues to be to slam the hard facts on the table and to indicate the possible solutions to the problems".

H.-J. Schellnhuber in *Frankfurter Allgemeine* from 2012-06-19

Interdisciplinary research

Different concerns: scientific, economic, political, ethical . . .

Different methodologies: empirical, simulations, Gedankenexperimente, stakeholder dialogues, participatory games . . .

Different specialized languages

Common ground: English, Mathematics, popular culture (e.g., "Lord of the Rings").

Focal concepts: can be intuitively understood by all and structure discourse. Example: vulnerability.

Vulnerability

. . . The complexity of the climate, ecological, social and economic systems that researchers are modelling means that the validity of scenario results will inevitably be subject to ongoing criticism.

. . . What this criticism does, however, is emphasize the need for a strong foundation upon which scenarios (*i.e., modelling*) can be applied, a foundation that provides a basis for managing risk despite uncertainties associated with future climate changes.

This foundation lies in the concept of vulnerability.

(from "Climate Change Impacts and Adaptation", 2004)

Definitions of *vulnerability*

"...a human condition or process resulting from physical, social and environmental factors which determine the likelihood and damage from the impact of a given hazard" (UNDP Annual Report, 2004)

"Vulnerability [...] is a way of conceptualizing what may happen to an identifiable population under conditions of particular risk and hazards." (Cannon et al. 2004)

"...the degree to which a system is susceptible to and unable to cope with, adverse effects of climate change, including climate variability and extremes. " (The Intergovernmental Panel on Climate Change, 2007)

...many, many others (e.g., Thywissen 2006 lists 32 more!)

The OED definition

vulnerable (adj.):

1. exposed to the possibility of being attacked or harmed, either physically or emotionally: *we were in a vulnerable position* | *small fish are vulnerable to predators*

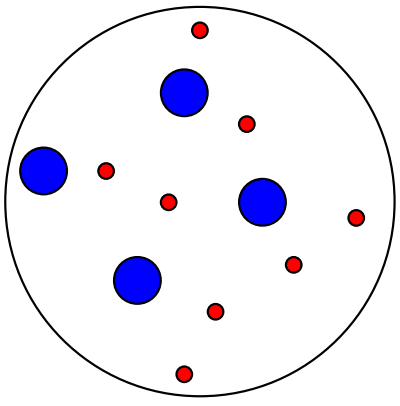
2. Bridge (of a partnership) liable to higher penalties, either by convention or through having won one game towards a rubber.

(Oxford English Dictionary, 2005)

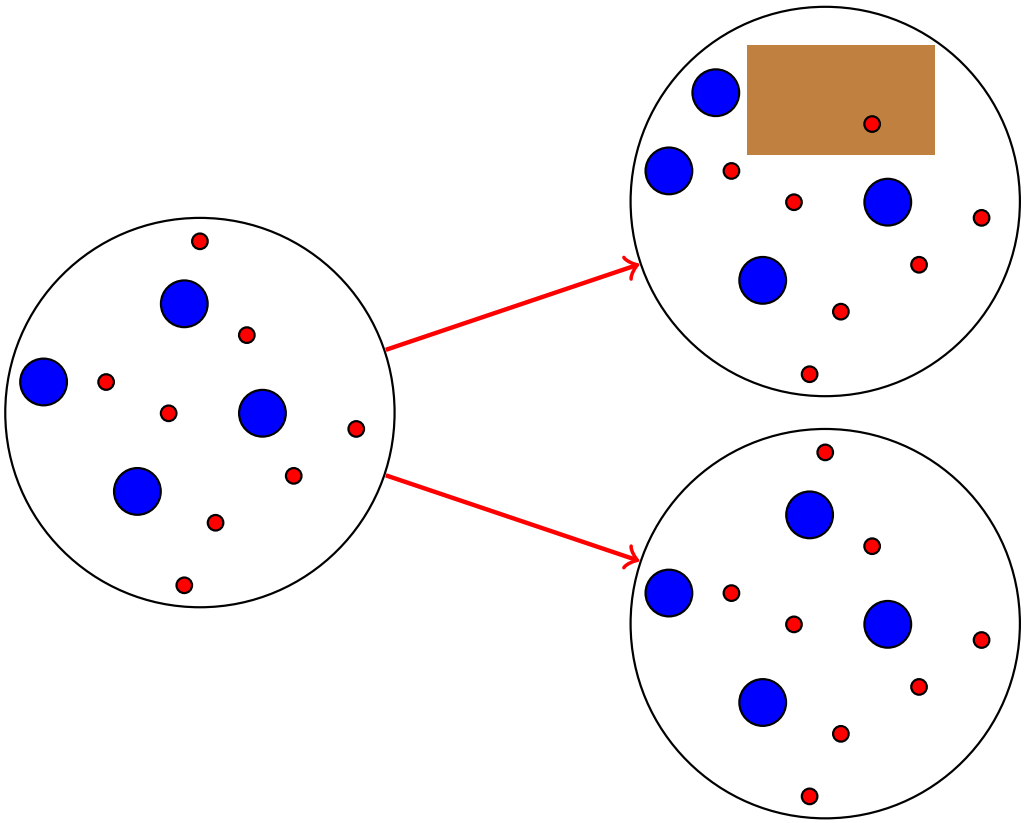
The practice of vulnerability assessment

1. run model a number of times under various scenarios
2. collect "indicators" of "badness" along the different trajectories (e.g. GDP loss, average temperature increase, lost lives . . .)
3. aggregate the collected data (in most cases by some sort of weighted average) into a vulnerability measure

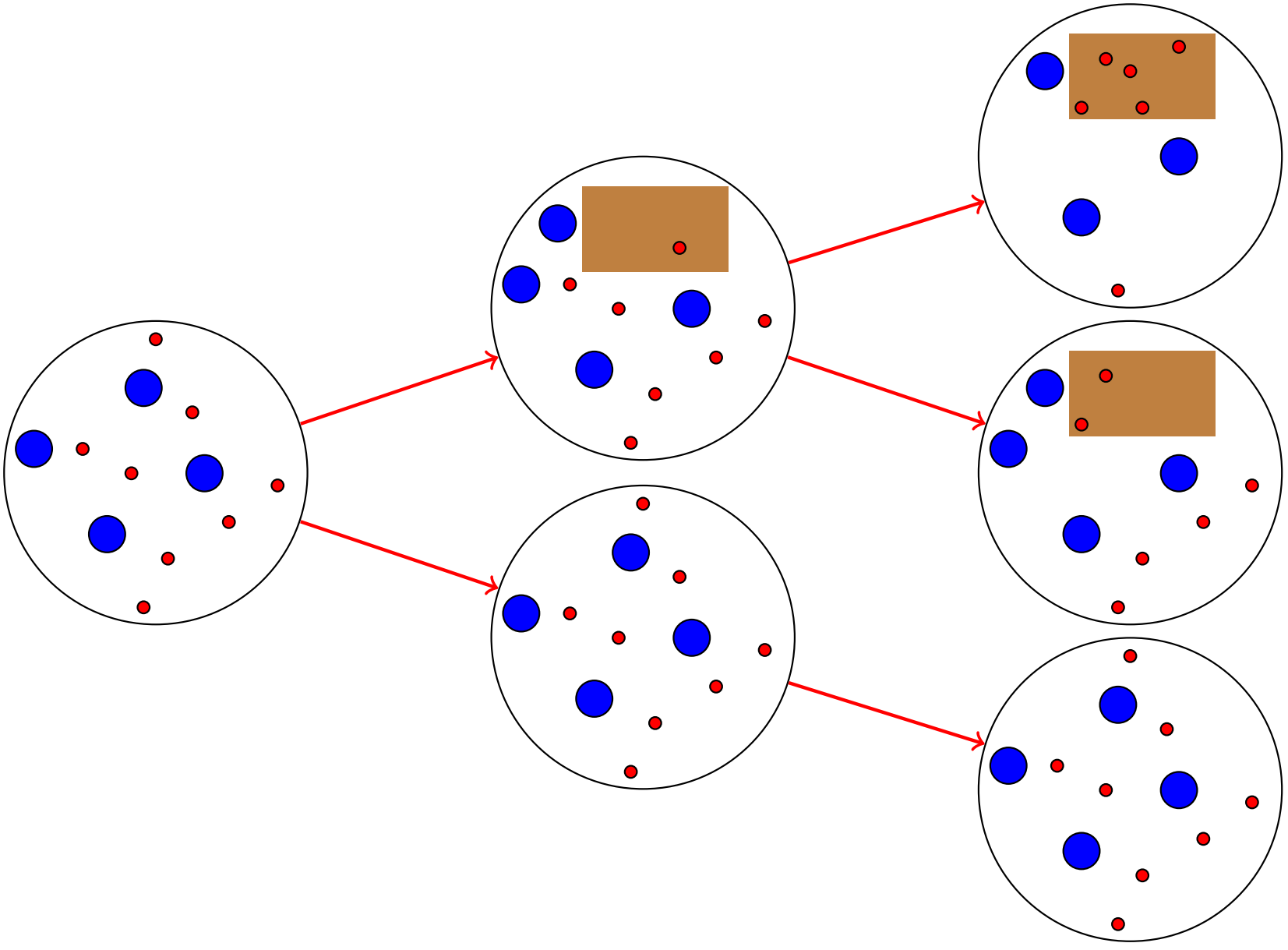
Example: scenarios



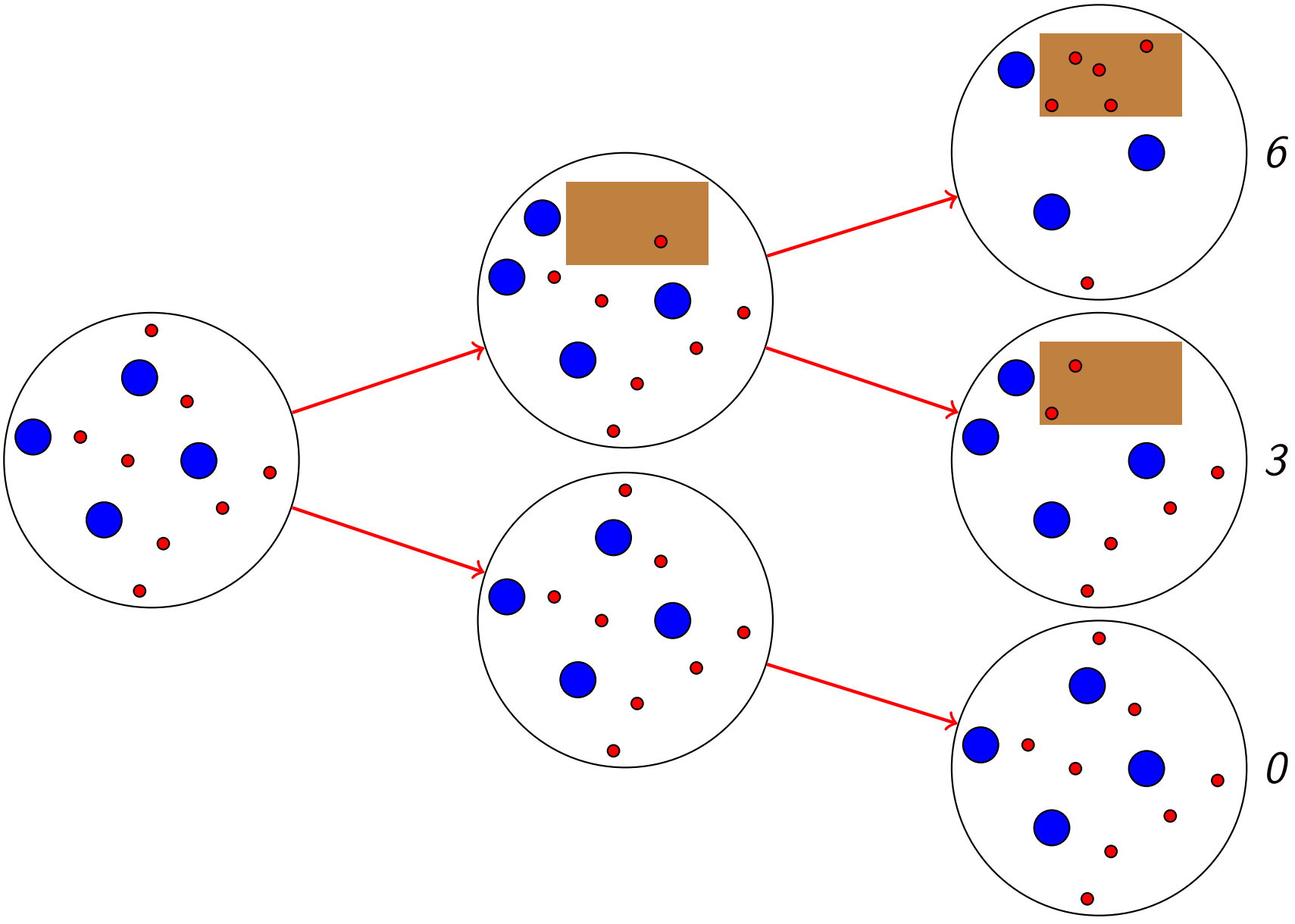
Example: scenarios



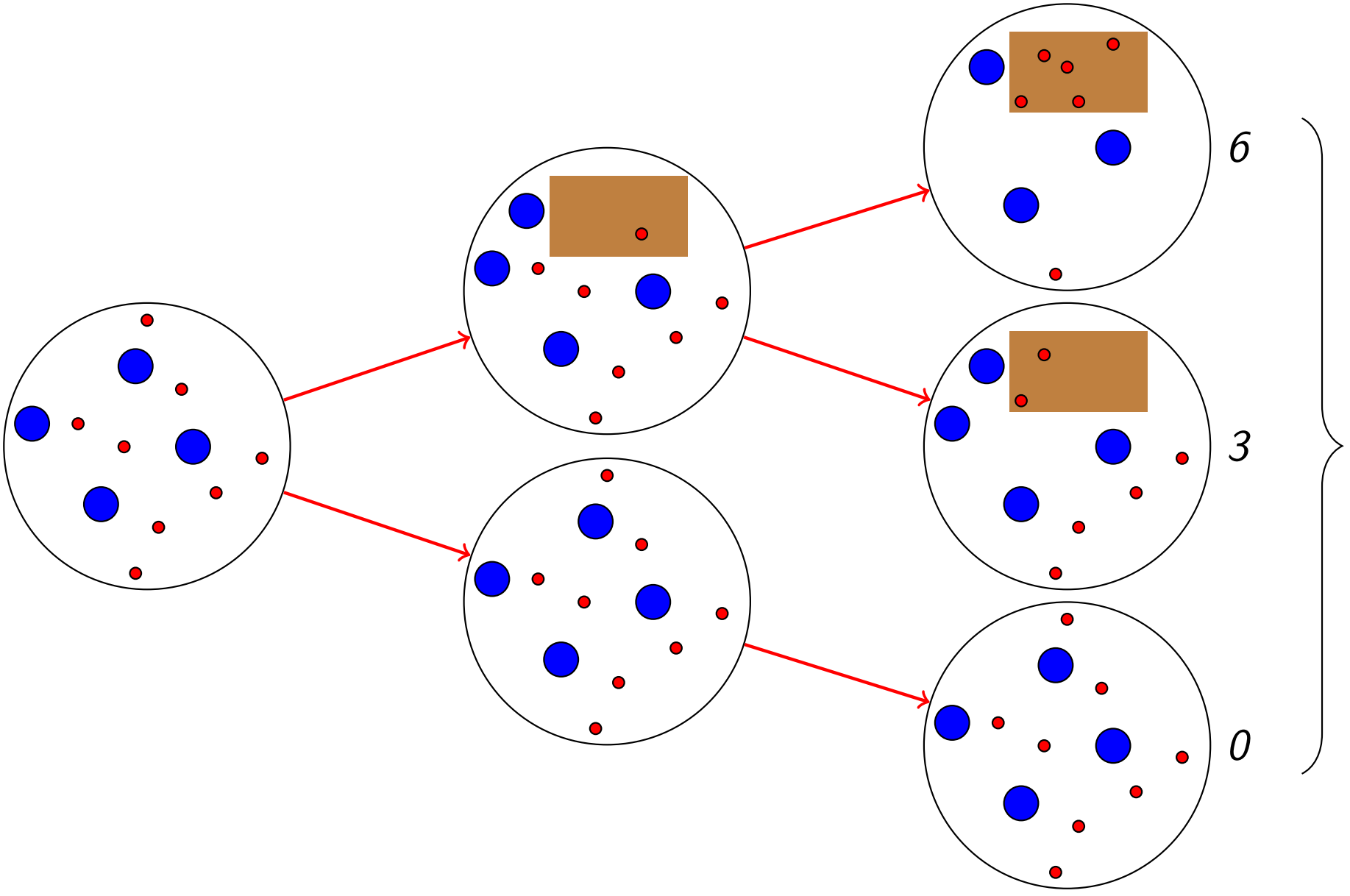
Example: scenarios



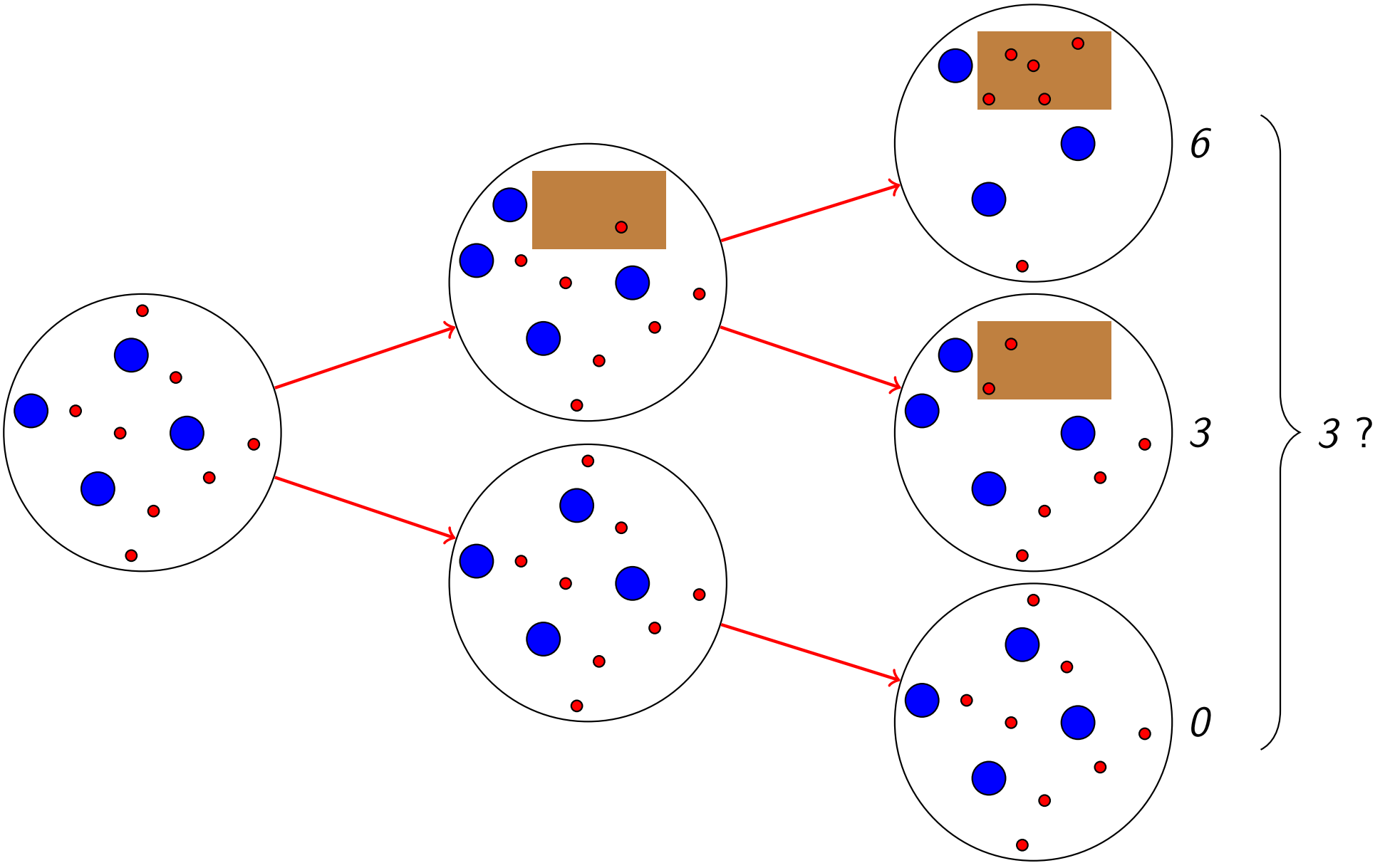
Example: scenarios



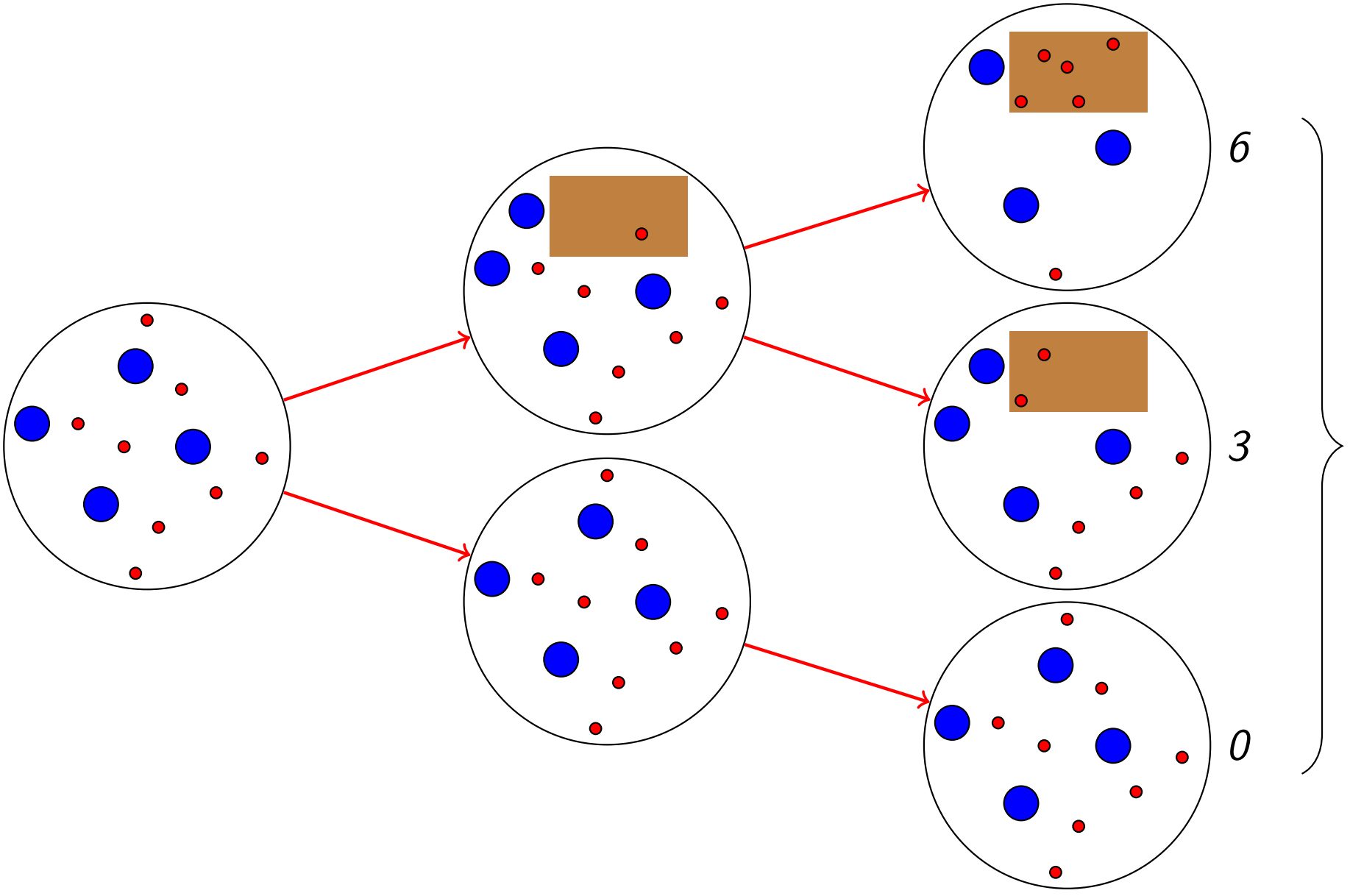
Example: scenarios



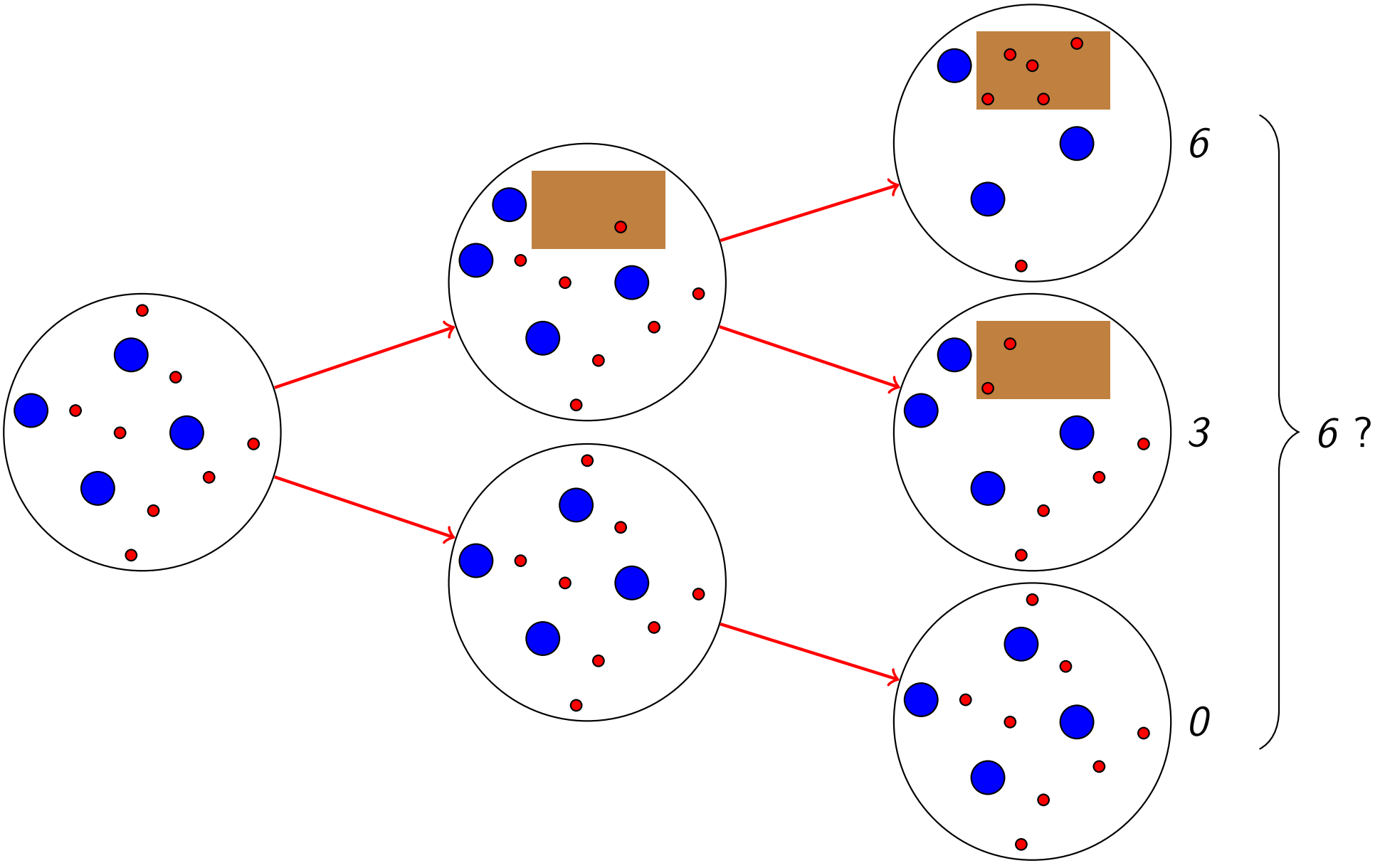
Example: scenarios



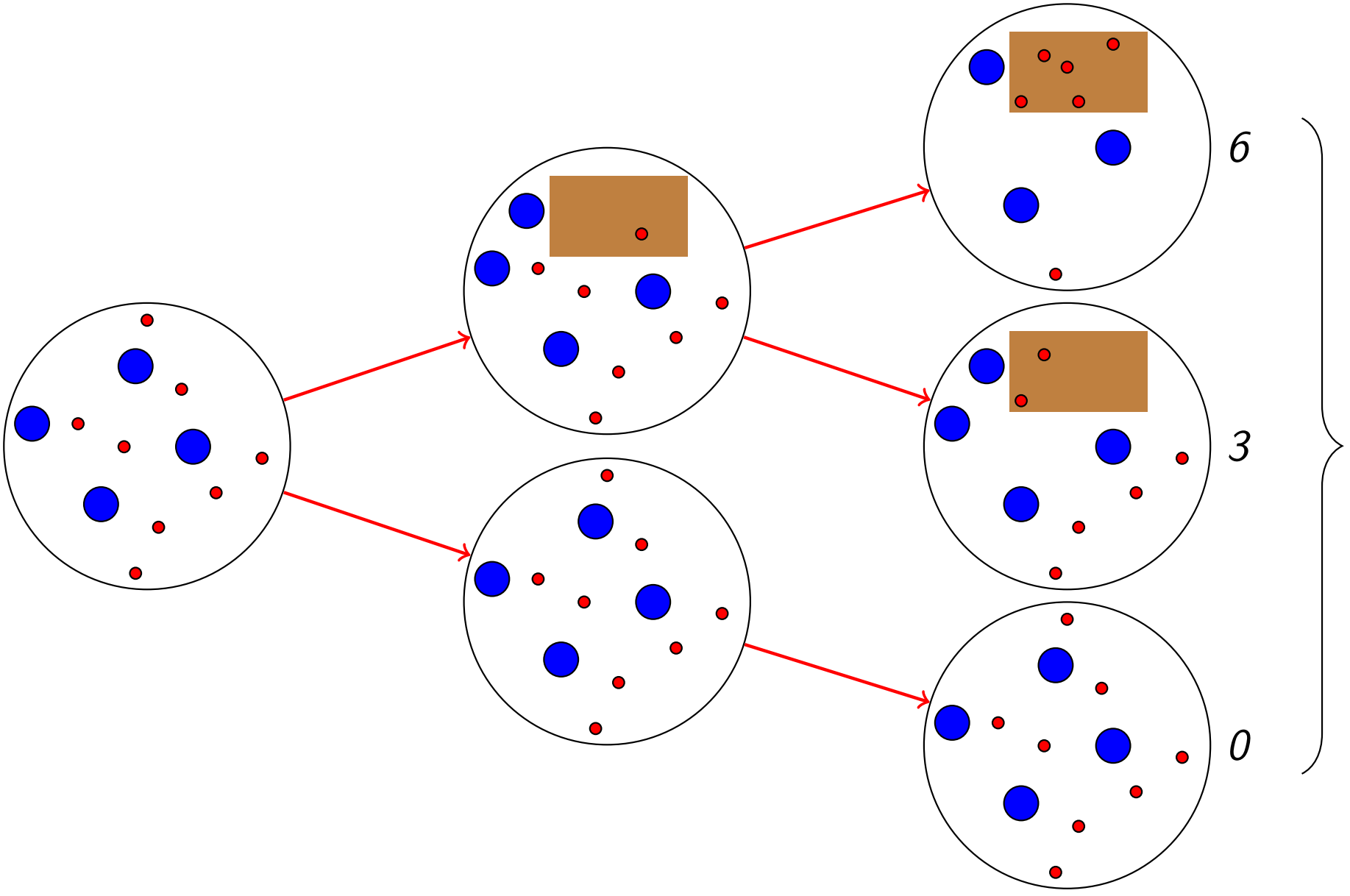
Example: scenarios



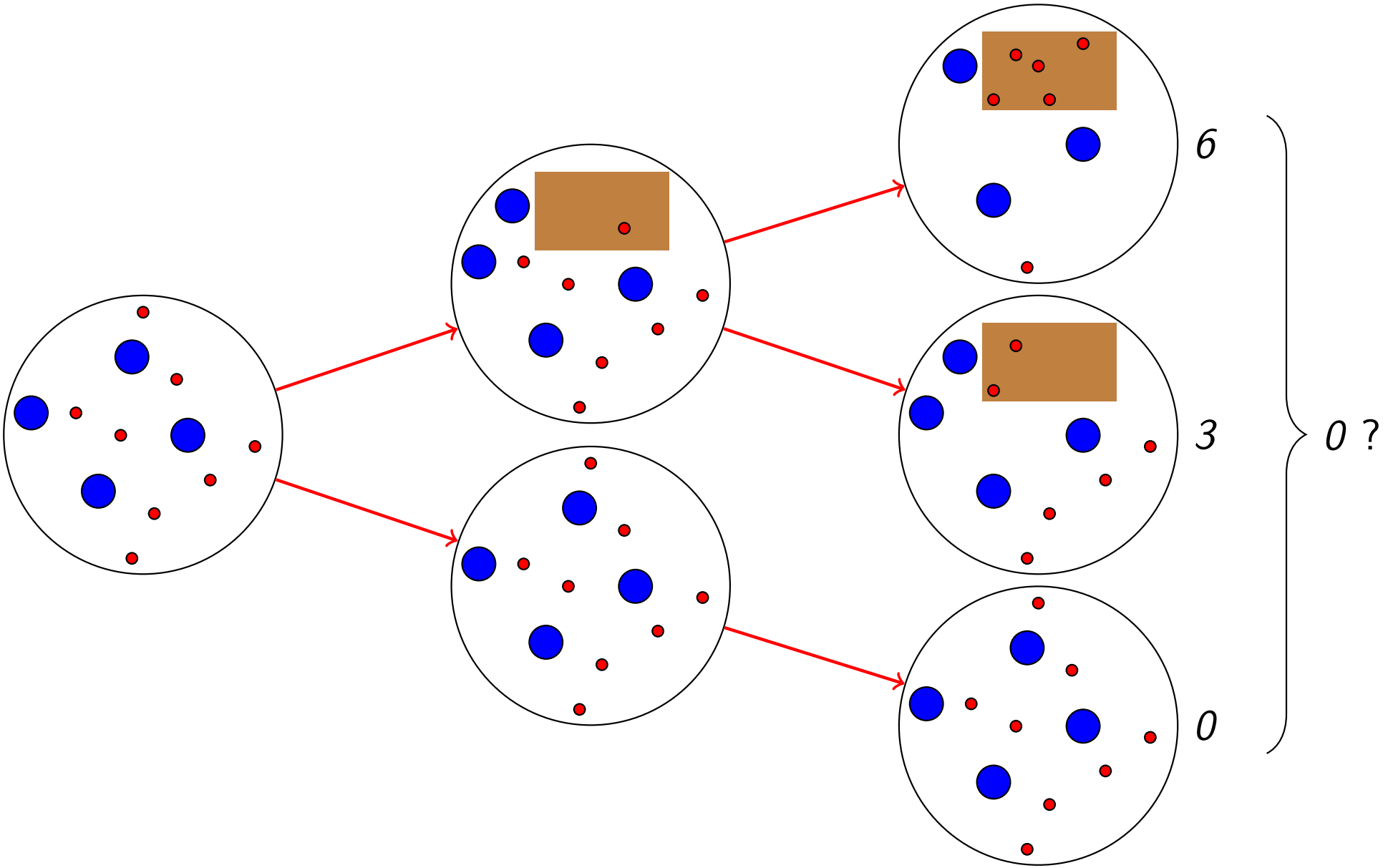
Example: scenarios



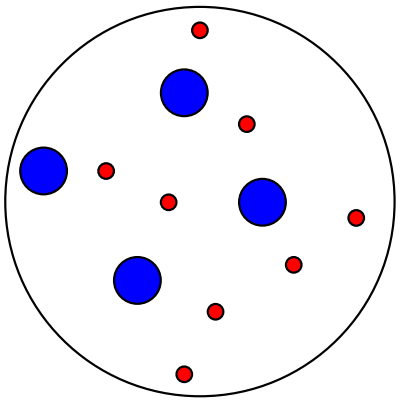
Example: scenarios



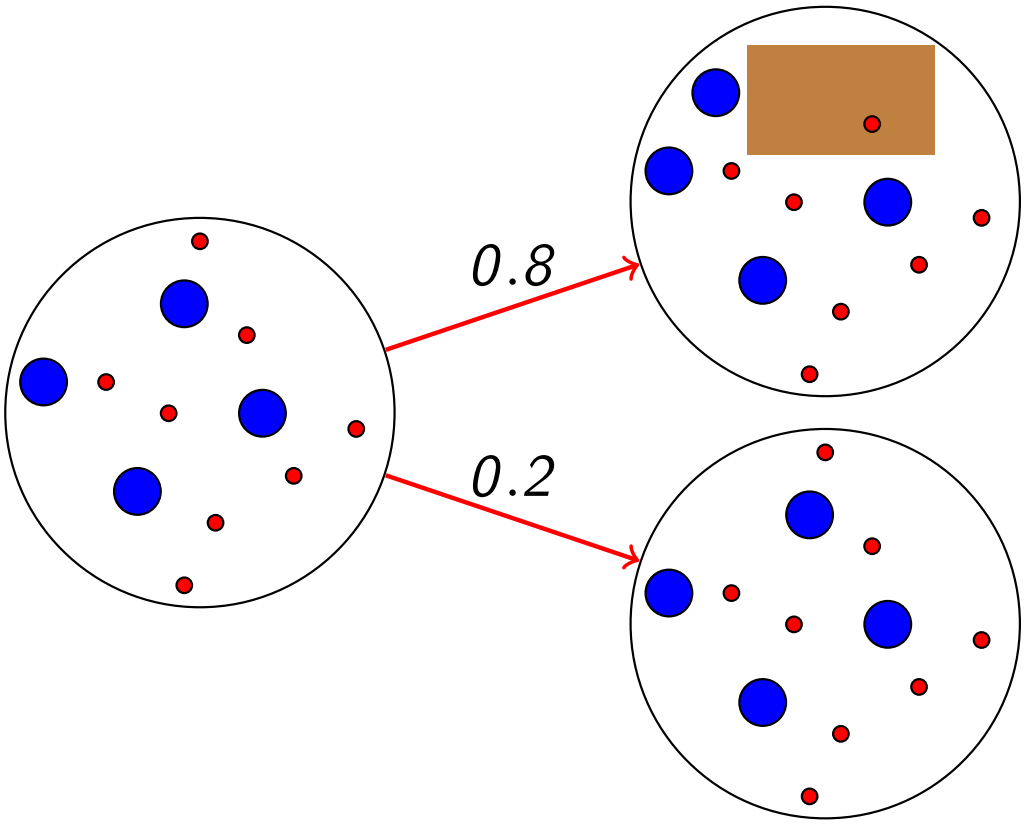
Example: scenarios



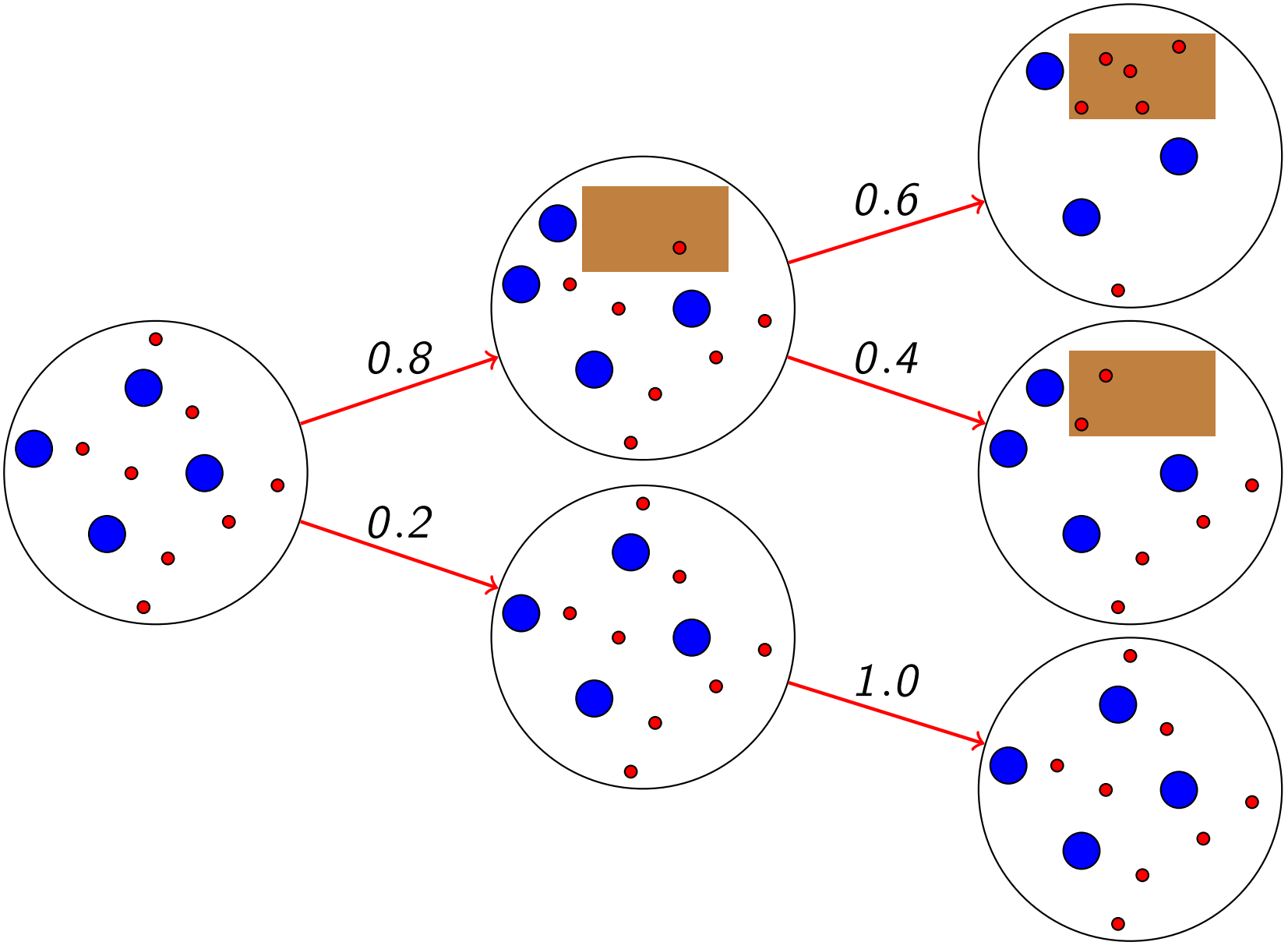
Example: stochastic system



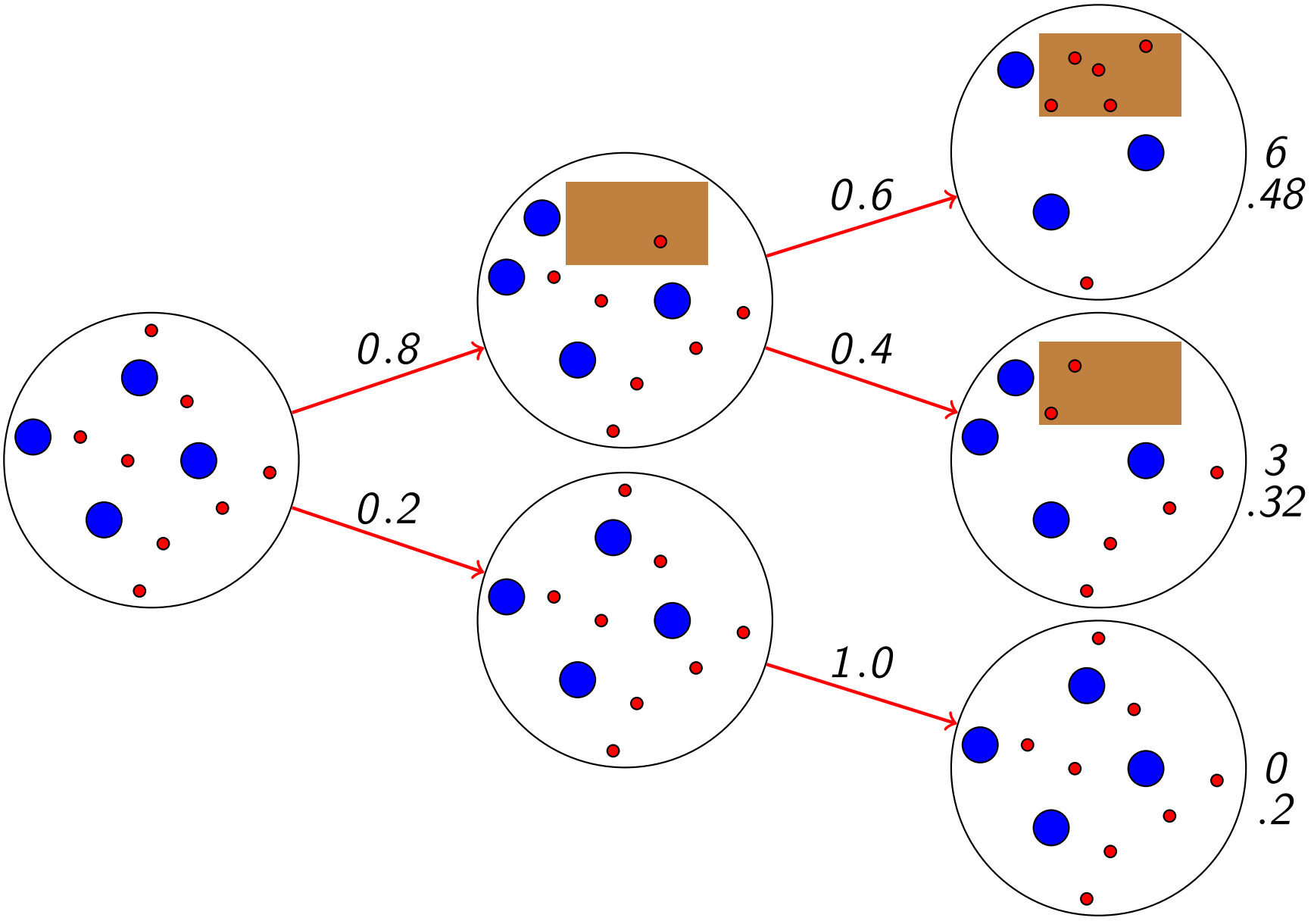
Example: stochastic system



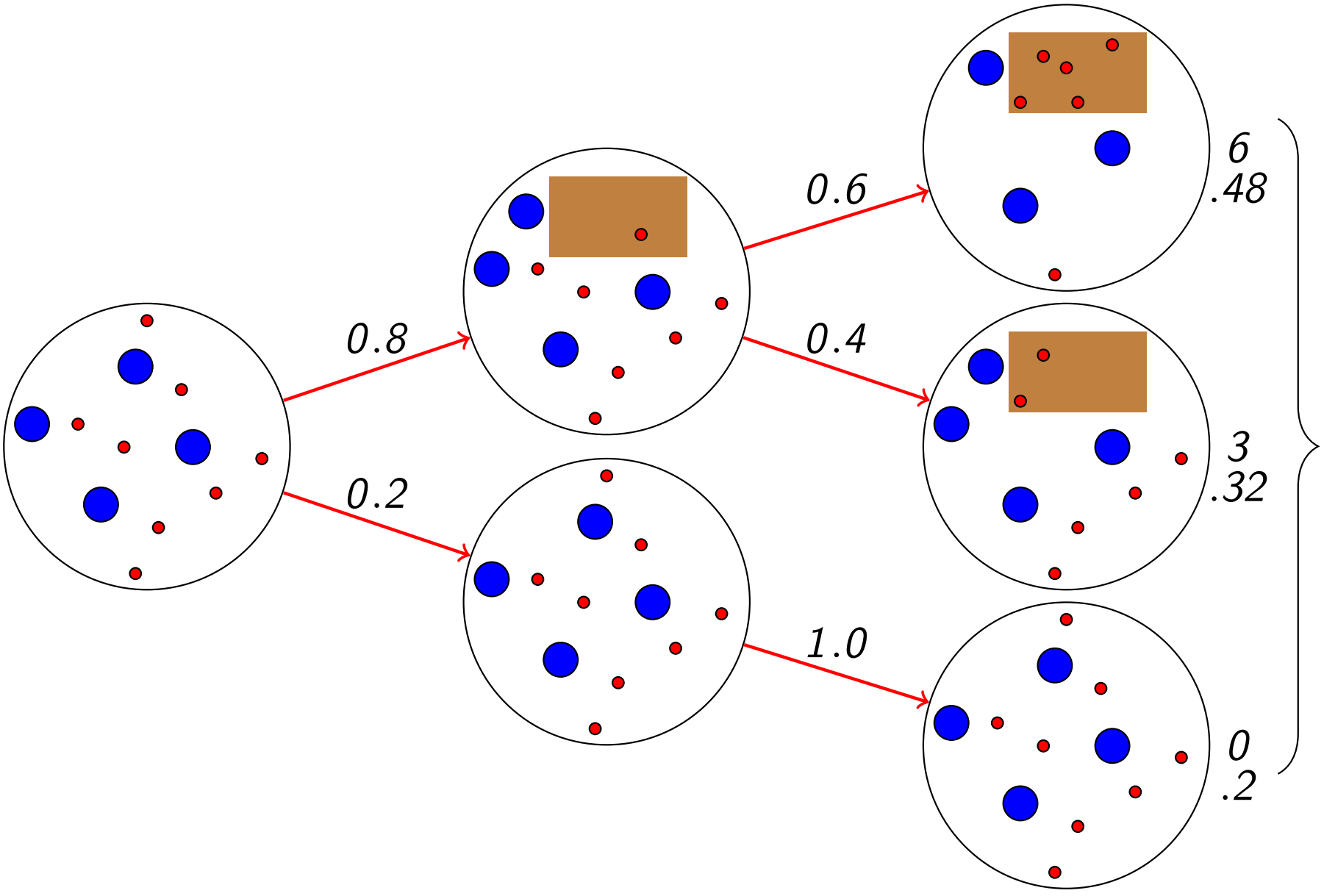
Example: stochastic system



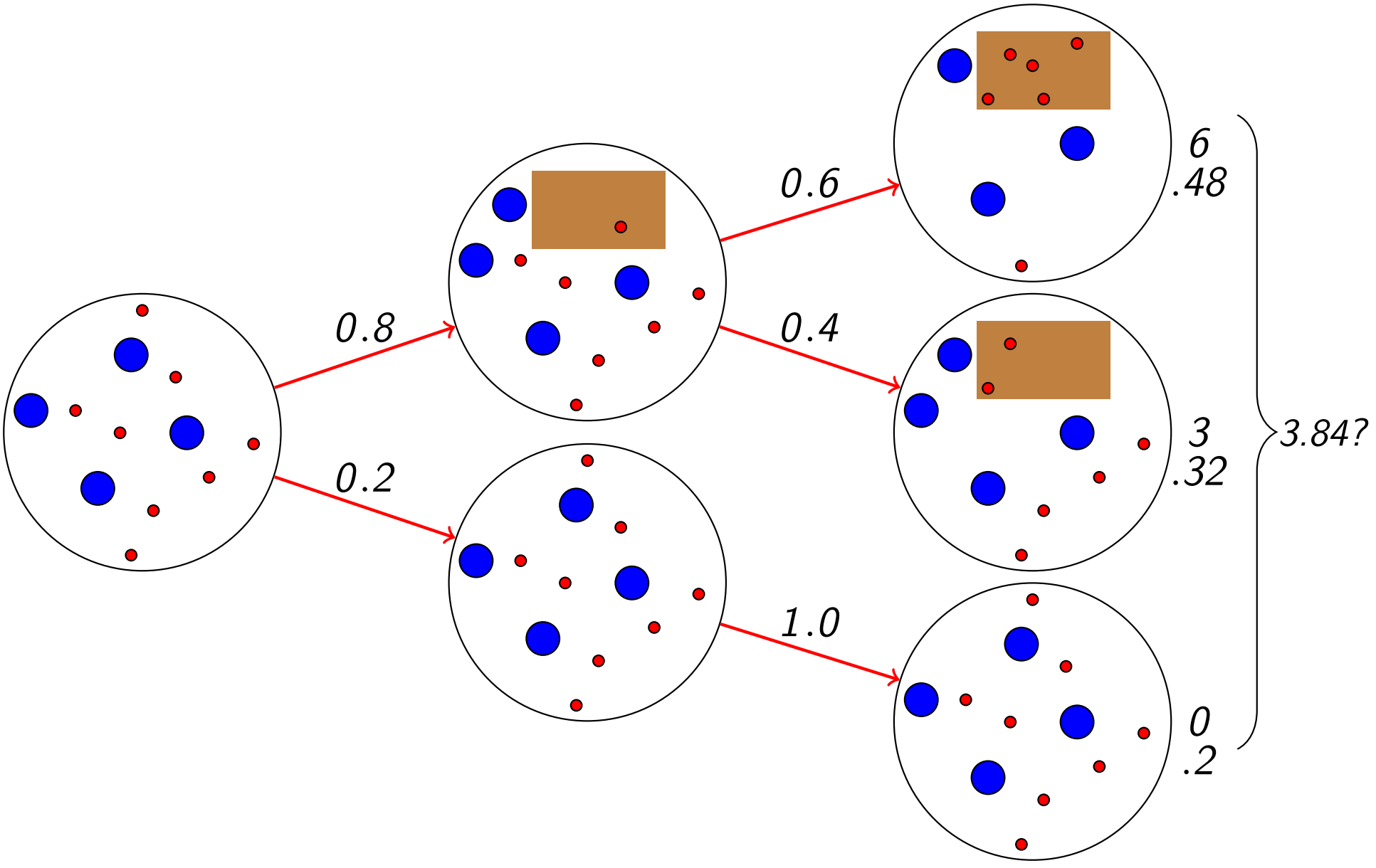
Example: stochastic system



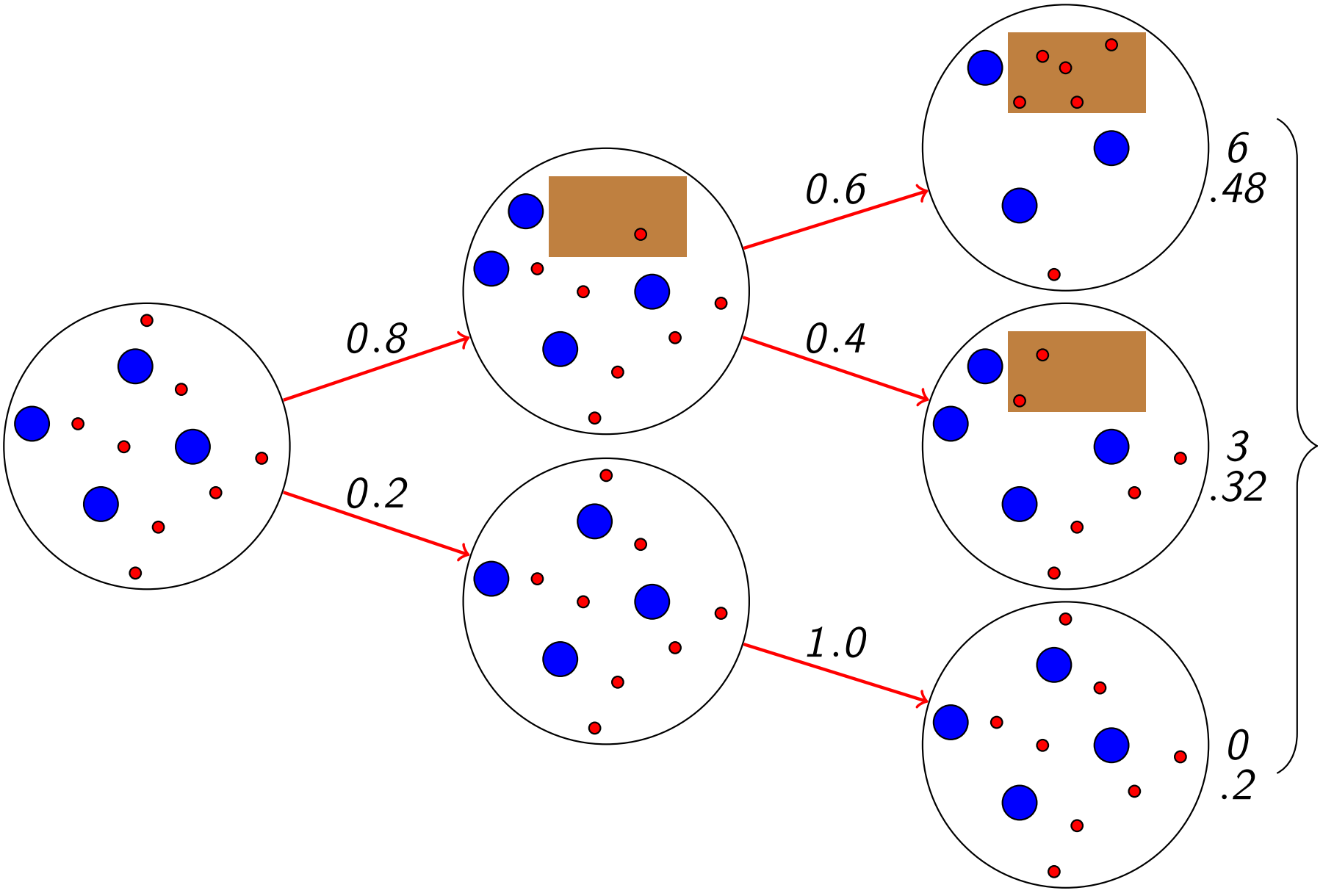
Example: stochastic system



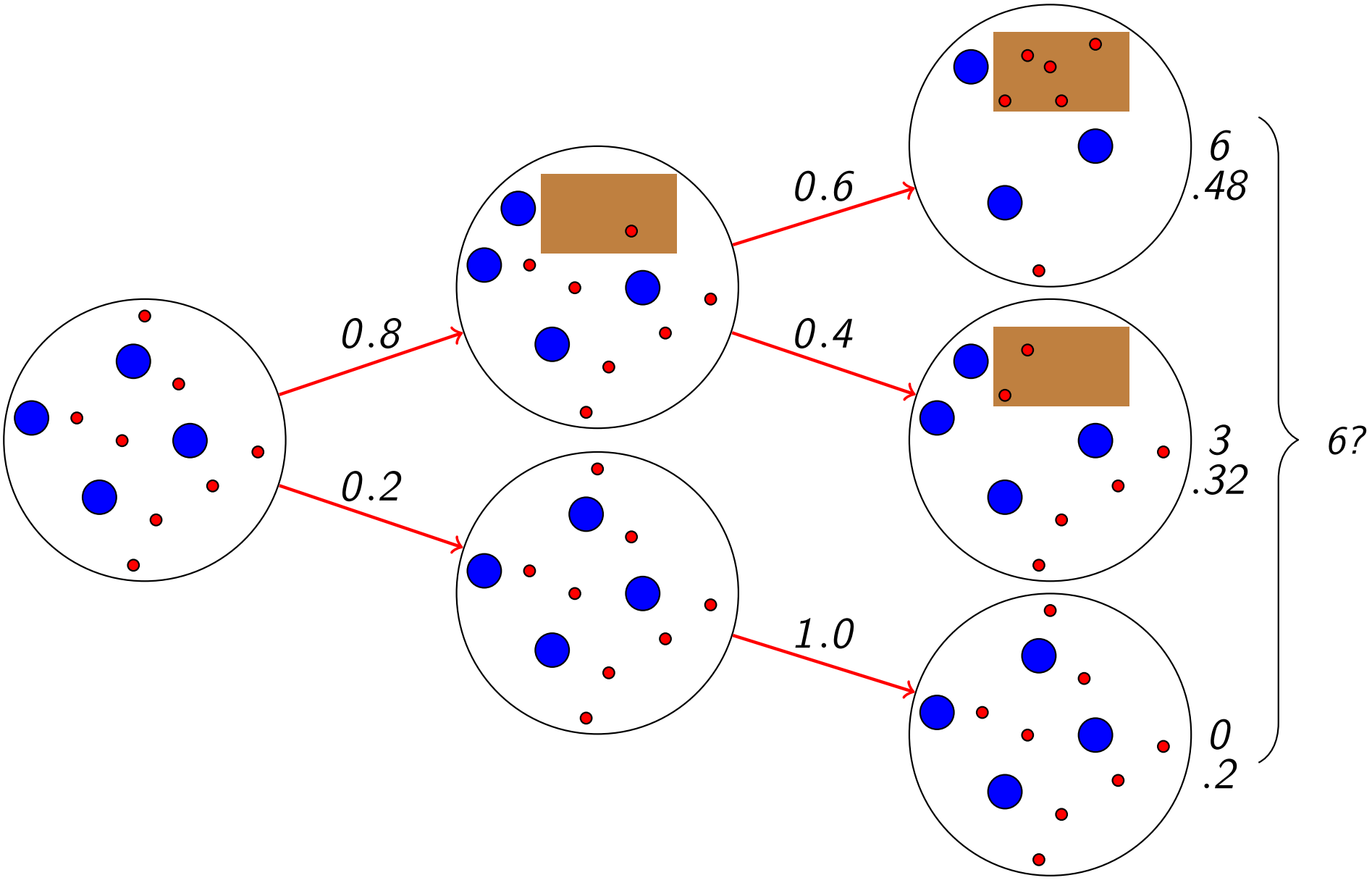
Example: stochastic system



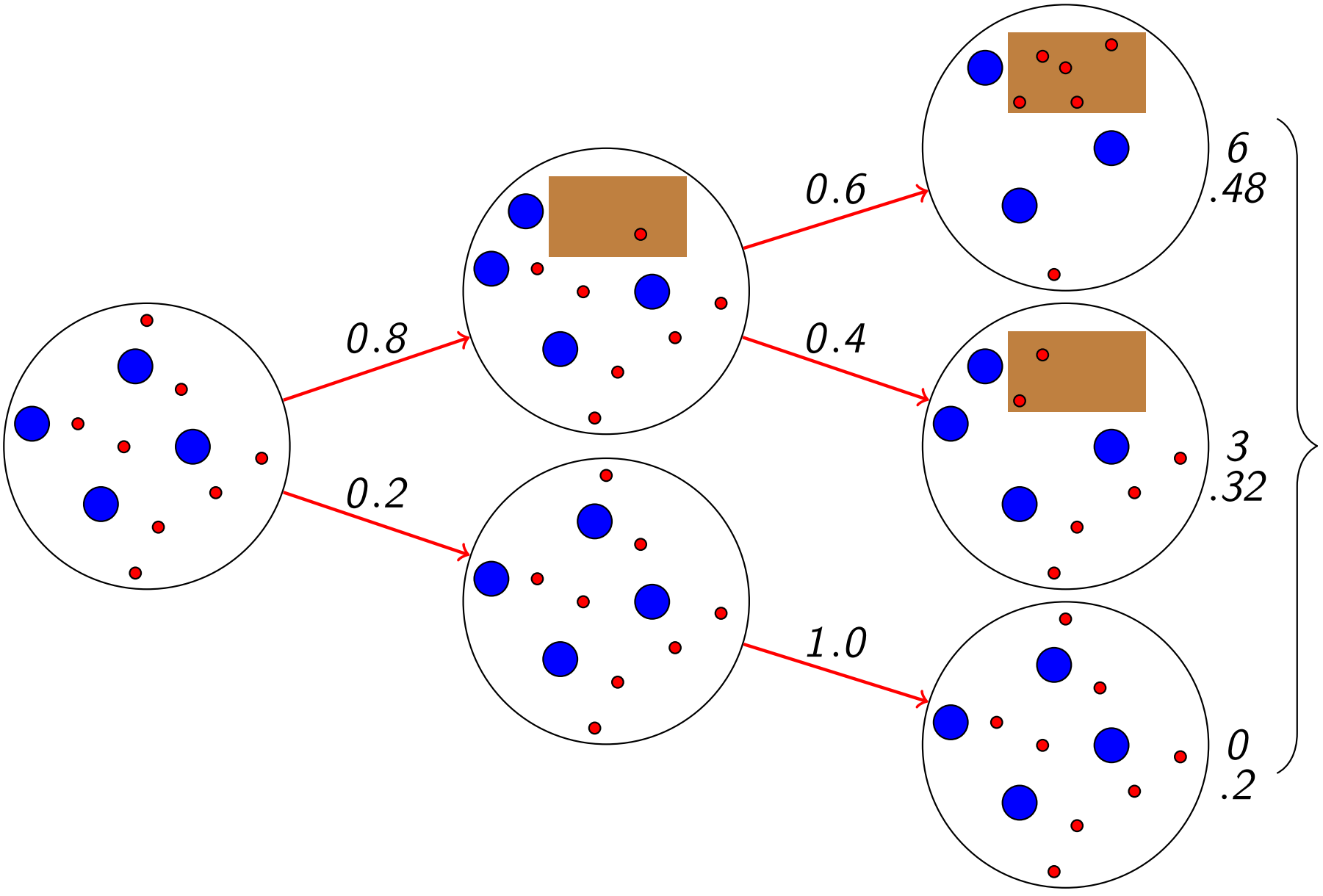
Example: stochastic system



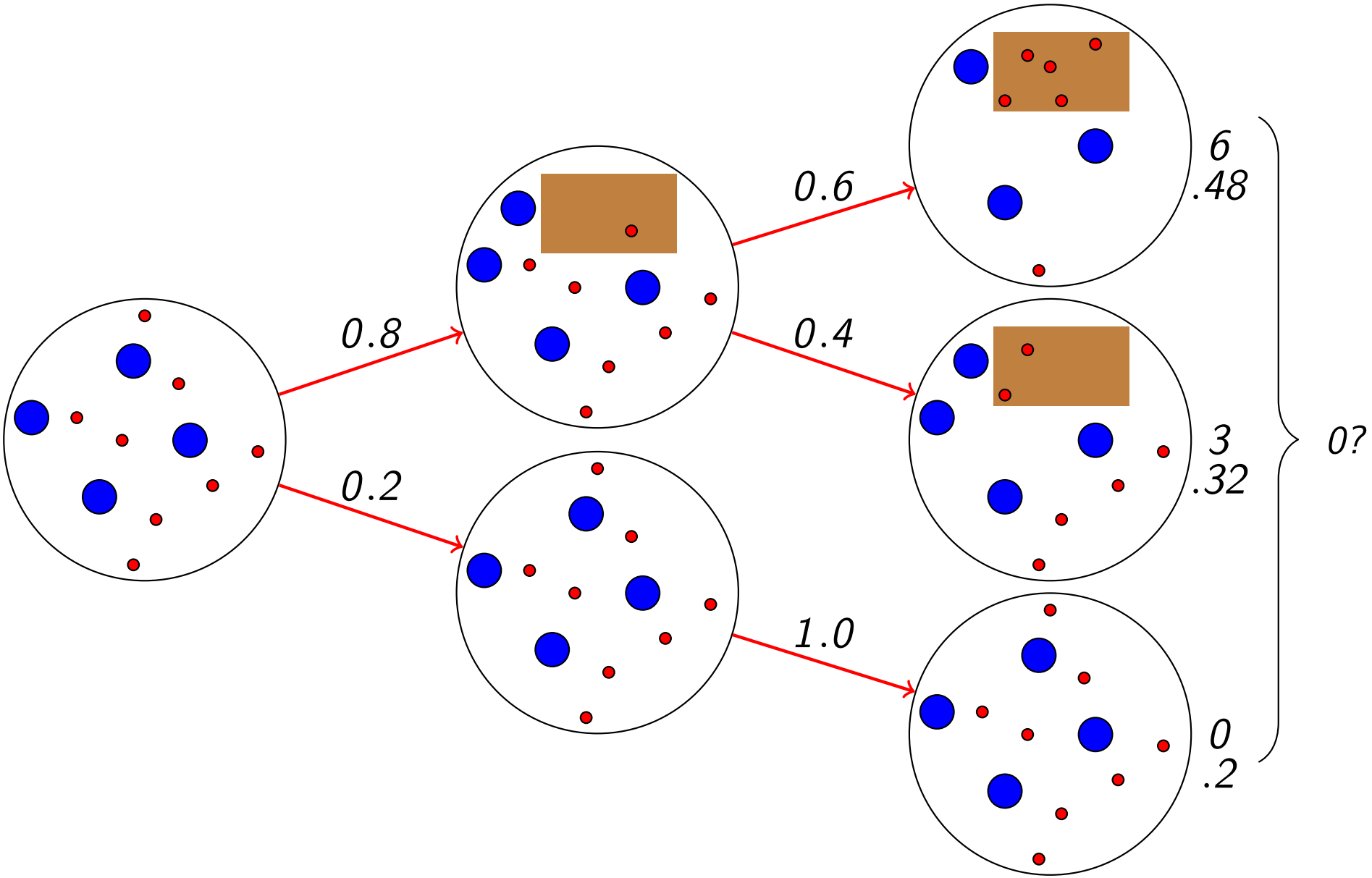
Example: stochastic system



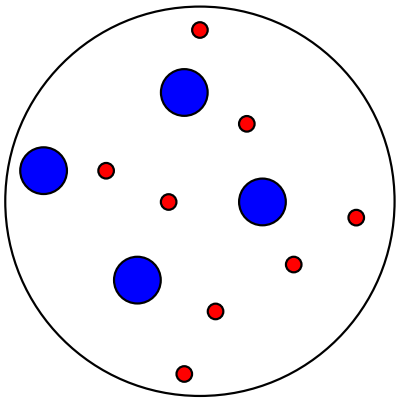
Example: stochastic system



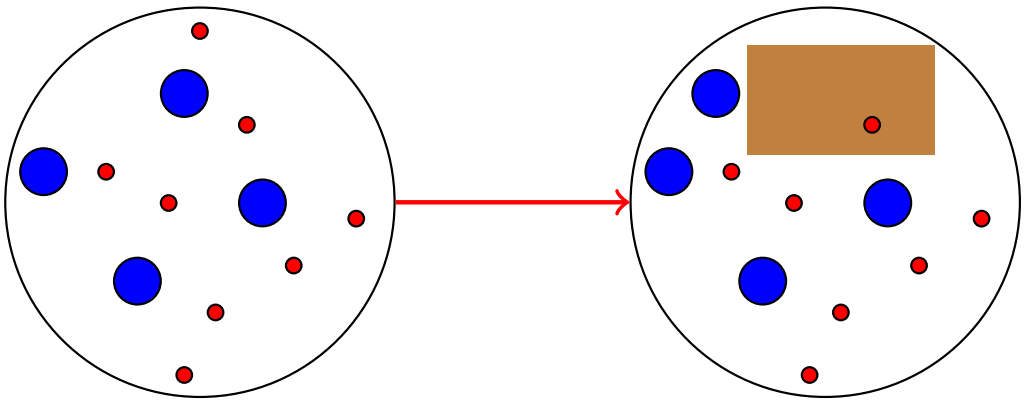
Example: stochastic system



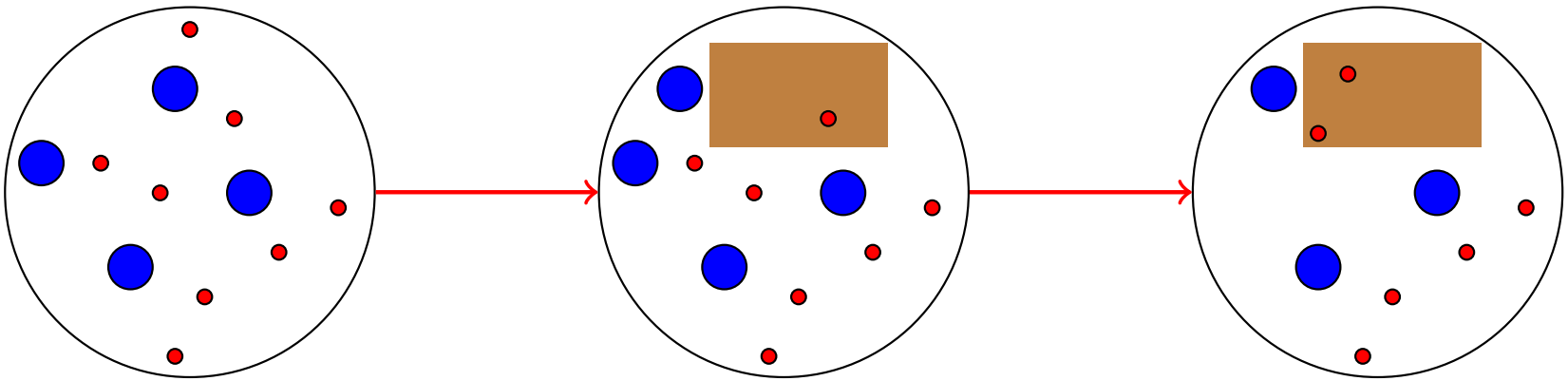
Example: deterministic system



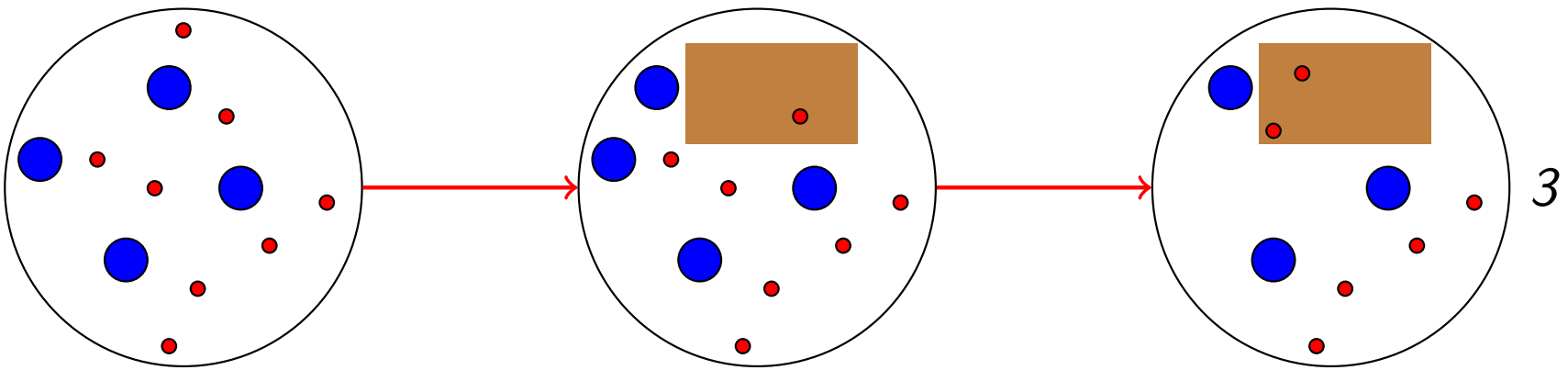
Example: deterministic system



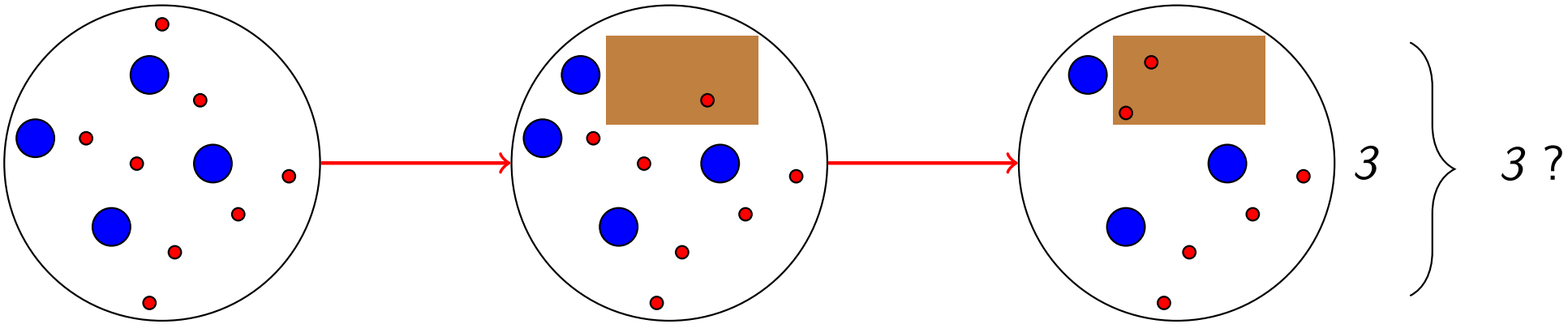
Example: deterministic system



Example: deterministic system



Example: deterministic system



Formalisation

$$States \ : \ Set$$

E.g.: $States \ = \ \{(nsmall, nbig, npred); nsmall, nbig, npred \ \in \ \mathbb{N}\}$

Formalisation

$$States \ : \ Set$$

E.g.: $States \ = \ \{(nsmall, nbig, npred); nsmall, nbig, npred \ \in \ \mathbb{N}\}$

$$Trajectory \ : \ List \ States$$

E.g., $trj \ = \ [\ (ns0, nb0, np0) \ , \ \ (ns1, nb1, np1) \ , \ \ ...]$

Formalisation

$$States \ : \ Set$$

E.g.: $States \ = \ \{(nsmall, nbig, npred); nsmall, nbig, npred \ \in \ \mathbb{N}\}$

$$Trajectory \ : \ List \ States$$

E.g., $trj \ = \ [\ (ns0, nb0, np0) \ , \ \ (ns1, nb1, np1) \ , \ \ ...]$

$$scenarios \ : \ State \ \rightarrow \ List \ Trajectory$$

E.g., $scenarios \ (ns0, nb0, np0) \ = \ [trj0, trj1, trj2]$

Formalisation

$$States \ : \ Set$$

E.g.: $States \ = \ \{(nsmall, nbig, npred); nsmall, nbig, npred \ \in \ \mathbb{N}\}$

$$Trajectory \ : \ List \ States$$

E.g., $trj \ = \ [\ (ns0, nb0, np0) \ , \ \ (ns1, nb1, np1) \ , \ \ ...]$

$$scenarios \ : \ State \ \rightarrow \ List \ Trajectory$$

E.g., $scenarios \ (ns0, nb0, np0) \ = \ [trj0, trj1, trj2]$

$$stochastic \ : \ State \ \rightarrow \ Prob \ Trajectory$$

E.g.,
$stochastic \ (ns0, nb0, np0) \ = \ [\ (trj0, p0) \ , \ \ (trj1, p1) \ , \ \ (trj2, p2) \ ]$

Formalisation

$$States \ : \ Set$$

E.g.: $States \ = \ \{(nsmall, nbig, npred); nsmall, nbig, npred \ \in \ \mathbb{N}\}$

$$Trajectory \ : \ List \ States$$

E.g., $trj \ = \ [\ (ns0, nb0, np0) \ , \ \ (ns1, nb1, np1) \ , \ \ ...]$

$$scenarios \ : \ State \ \rightarrow \ List \ Trajectory$$

E.g., $scenarios \ (ns0, nb0, np0) \ = \ [trj0, trj1, trj2]$

$$stochastic \ : \ State \ \rightarrow \ Prob \ Trajectory$$

E.g.,
$stochastic \ (ns0, nb0, np0) \ = \ [\ (trj0, p0) \ , \ \ (trj1, p1) \ , \ \ (trj2, p2) \ ]$

$$deterministic \ : \ State \ \rightarrow \ Trajectory$$

E.g., $deterministic \ (ns0, nb0, np0) \ = \ trj$

Formalisation

$States : Set$

E.g.: $States = \{(nsmall, nbig, npred); nsmall, nbig, npred \in \mathbb{N}\}$

$Trajectory : List\ States$

E.g., $trj = [\ (ns0, nb0, np0)\ ,\ \ (ns1, nb1, np1)\ ,\ \ ...]$

$scenarios : State \rightarrow List\ Trajectory$

E.g., $scenarios\ (ns0, nb0, np0) = [trj0, trj1, trj2]$

$stochastic : State \rightarrow Prob\ Trajectory$

E.g.,
$stochastic\ (ns0, nb0, np0) = [\ (trj0, p0)\ ,\ \ (trj1, p1)\ ,\ \ (trj2, p2)\ ]$

$deterministic : State \rightarrow Trajectory$

E.g., $deterministic\ (ns0, nb0, np0) = trj$

Formalisation

$$States \ : \ Set$$

E.g.: $States \ = \ \{(nsmall, nbig, npred); nsmall, nbig, npred \ \in \ \mathbb{N}\}$

$$Trajectory \ : \ List \ States$$

E.g., $trj \ = \ [\ (ns0, nb0, np0) \ , \ \ (ns1, nb1, np1) \ , \ \ ...]$

$$possible \ : \ State \ \rightarrow \ F \ Trajectory$$

Formalisation

$$harm : Trajectory \rightarrow V$$

Formalisation

$$harm : Trajectory \rightarrow V$$

E.g., for scenarios:

$$[harm\ trj0, harm\ trj1, harm\ trj2]$$

Formalisation

$$harm \ : \ Trajectory \ \rightarrow \ V$$

E.g., for scenarios:

$$[harm\ trj0, harm\ trj1, harm\ trj2]$$

E.g., for stochastic:

$$[harm\ (trj0, p0)\ ,\ \ harm\ (trj1, p1)\ ,\ \ harm\ (trj2, p2)\ ]$$

Formalisation

$$harm \ : \ Trajectory \ \rightarrow \ V$$

E.g., for scenarios:

$$[harm\ trj0, harm\ trj1, harm\ trj2]$$

E.g., for stochastic:

$$[harm\ (trj0, p0)\ ,\ \ harm\ (trj1, p1)\ ,\ \ harm\ (trj2, p2)\ ]$$

E.g., for deterministic:

$$harm\ trj$$

Formalisation

$$possible : State \rightarrow F\ Trajectory$$

$$harm : Trajectory \rightarrow V$$

Formalisation

$$possible : State \rightarrow F\ Trajectory$$

$$harm : Trajectory \rightarrow V$$

F **functor**, i.e., we can *map harm* inside the structure:

$$fmap\ harm \circ possible : State \rightarrow F\ V$$

Formalisation

$$possible \ : \ State \ \rightarrow \ F \ Trajectory$$

$$harm \ : \ Trajectory \ \rightarrow \ V$$

$$fmap \ harm \ \circ \ possible \ : \ State \ \rightarrow \ F \ V$$

Formalisation

$$possible : State \to F\ Trajectory$$

$$harm : Trajectory \to V$$

$$fmap\ harm \circ possible : State \to F\ V$$

We need to "measure" the resulting structure of harm values

$$measure : F\ V \to W$$

$$measure \circ fmap\ harm \circ possible$$

Formalisation

$$possible : State \to F\ Trajectory$$

$$harm : Trajectory \to V$$

$$fmap\ harm \circ possible : State \to F\ V$$

$$measure : F\ V \to W$$

$$vulnerability = measure \circ fmap\ harm \circ possible$$

Formalisation

$$vulnerability = measure \circ fmap\ harm \circ possible$$

How is this useful?

Formalisation

$$vulnerability = measure \circ fmap\ harm \circ possible$$

How is this useful?

All definitions in, e.g., Thywissen 2006, are special cases.

Therefore, we have a principled way of comparing them.

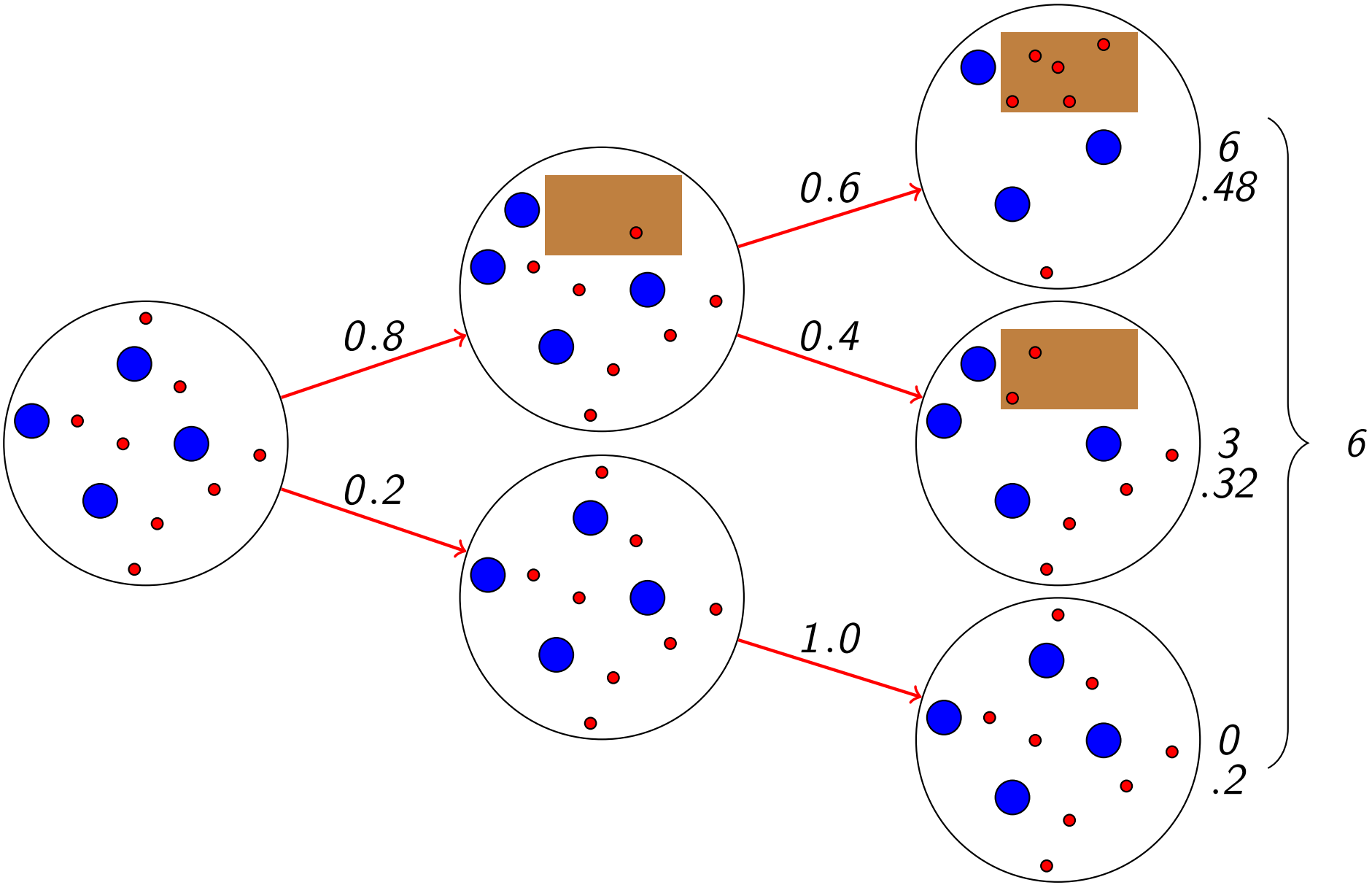
If we can find agreed upon conditions to impose on the components, we have a way of checking them.

Formalisation

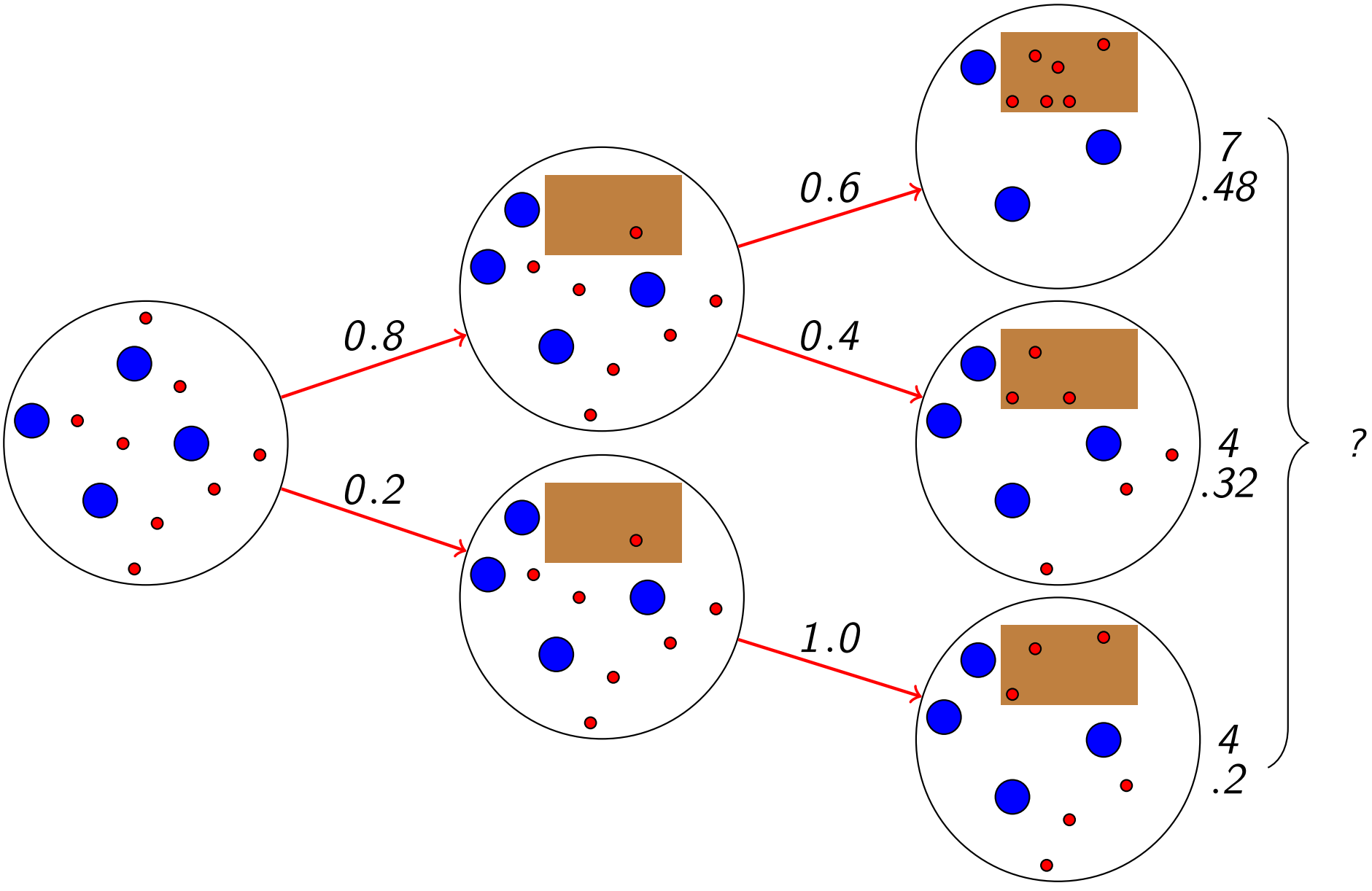
$$\mathit{vulnerability} = \mathit{measure} \circ \mathit{fmap}\ \mathit{harm} \circ \mathit{possible}$$

An agreed upon condition on *measure*: if in all trajectories the harm increases, the vulnerability must not decrease.

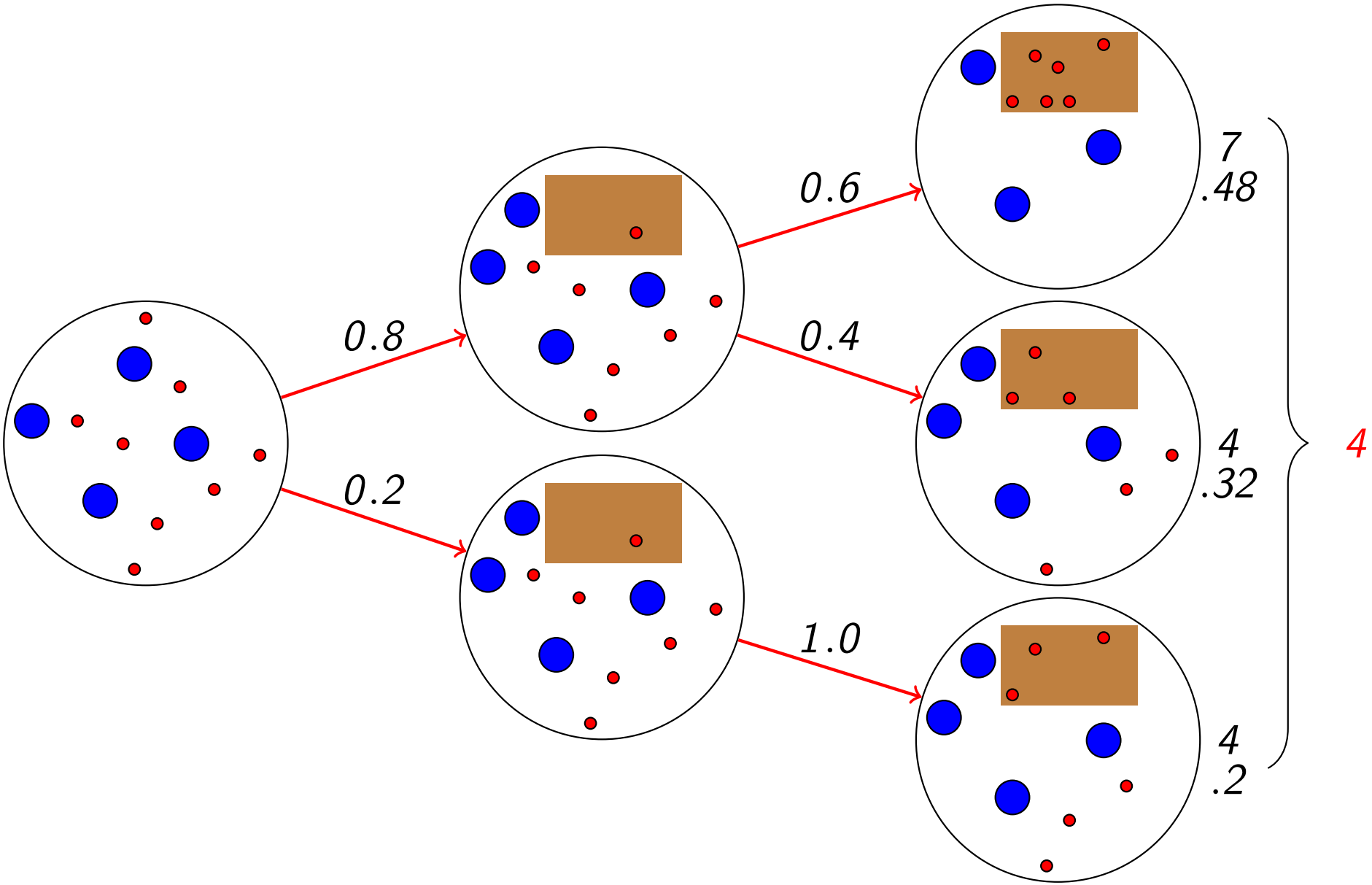
Example: monotonicity condition



Example: monotonicity condition



Example: monotonicity condition



Models of vulnerability

$$V = \int \left(\frac{\partial W/\partial X}{W/W_0}\right) P_X dX$$

where W "is a parabolic function of an independent variable X"

"Although it is impossible to determine the precise functional relationships [. . .] analysis based on simple theoretical models and multivariate regressions from empirical data can provide valuable information about critical relationships that can be applied in this measure."

Luers et al., Glob. Env. Change 2003

Models vs formalisation

The vast majority of models are like this: find a simple mathematical or computational system that is somehow similar to the "real" system.

Analyse the model and transfer the conclusions, *by analogy*, to the real system.

Analogy is the essence of modelling (and of applied mathematics).

What we have tried to do was to identify "ground rules" for analogies in the field of vulnerability to climate change.

Formalisation

Identifying "ground rules" for analogies in a field has much in common with grammatical investigations.

The aim is a description of current usage, making explicit, in mathematical terms, norms implied by this usage.

There are many sources of examples of "current usage", due to ubiquity of computer simulations in the social sciences. ("Simulation is a third way of doing science", Robert Axelrod, *Advancing the Art of Simulation in the Social Sciences*, 2003).

The future

- Systematically explore software models and reverse engineer specifications.
- Pay attention to recurring concepts, such as *preference, belief, fairness, trust, ...*
- Possible candidates: migration waves; relations of trust; responsibility in the context of automation; ...