Two odd things about computation

Daniel Murfet
University of Melbourne

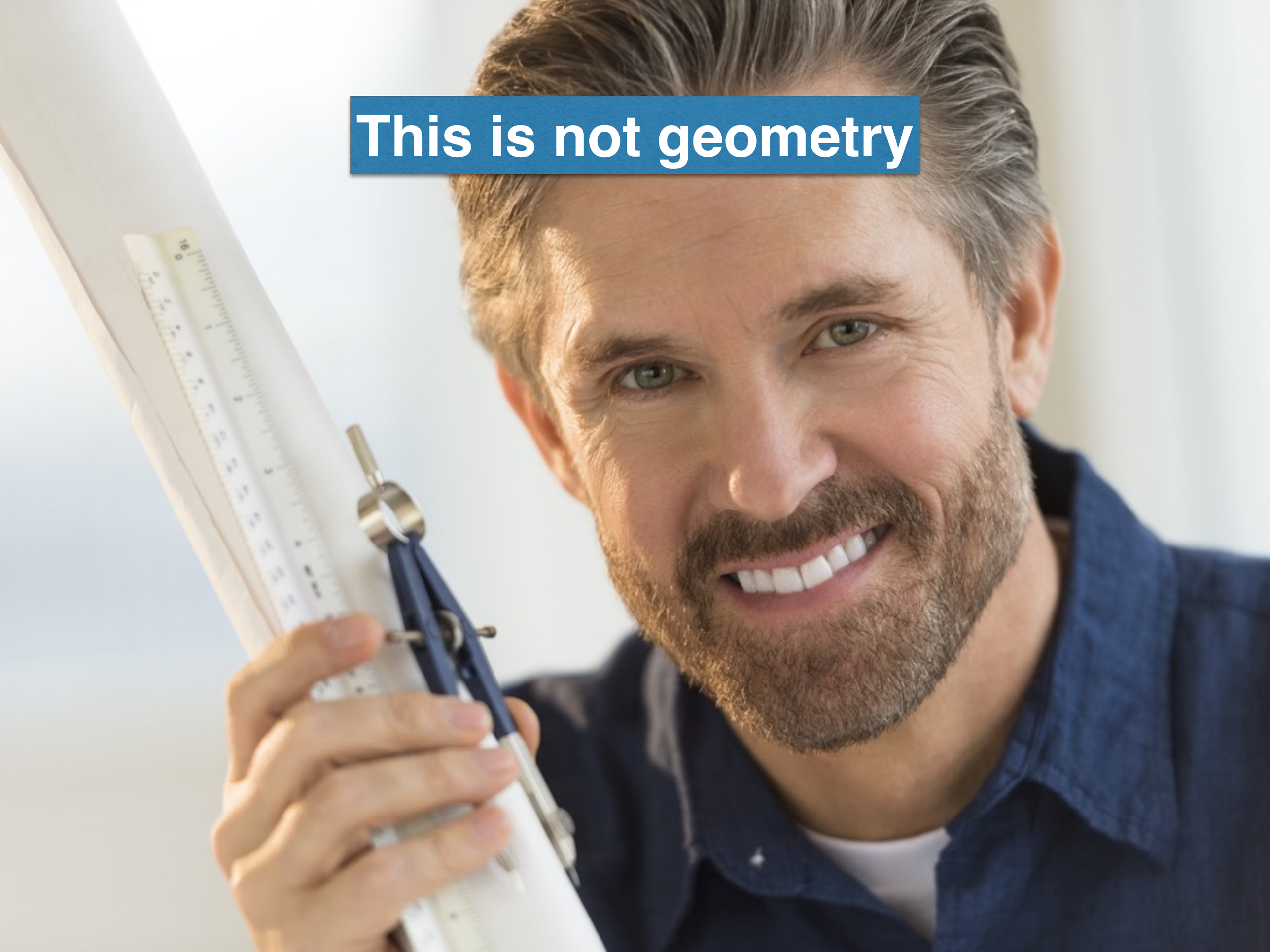
Two odd things

I. Maxwell's demon (1871)

Energy cost of computation

II. Russell's paradox (1901)

Time and space cost of computation


This is not geometry

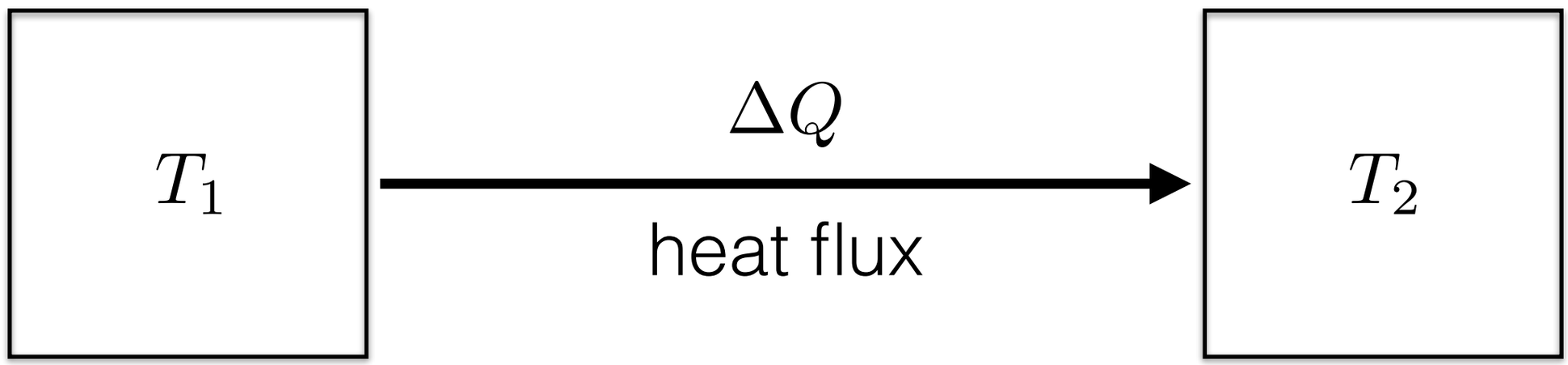
This is not computation

I. Maxwell's demon

How much energy must be used
in carrying out a computation?

Quasi-static flux of heat Q into a system at temperature T is associated with an increase of entropy S of that system.

$$\Delta Q = T\Delta S$$

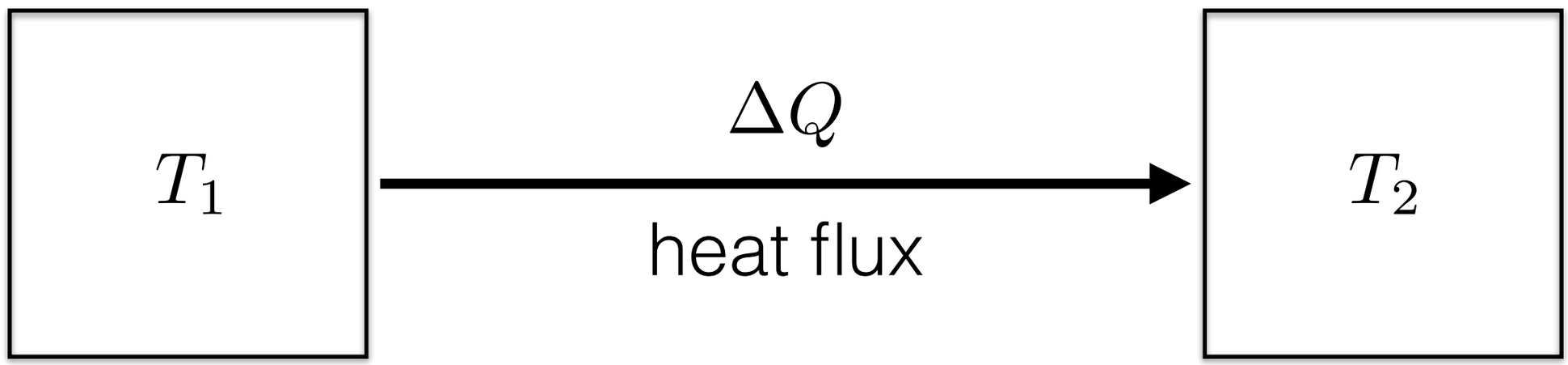


$$-T_1\Delta S_1 = \Delta Q = T_2\Delta S_2$$

$$\Delta S = \Delta S_2 + \Delta S_1 = \left(1 - \frac{T_2}{T_1}\right)\Delta S_2 \quad = \begin{cases} > 0 & T_1 > T_2 \\ < 0 & T_1 < T_2 \end{cases}$$

Second law: it is impossible to derive an engine which, working in a cycle, shall produce no effect other than the transfer of heat from a colder to a hotter body.

i.e. in a closed system always $\Delta S_{tot} \geq 0$



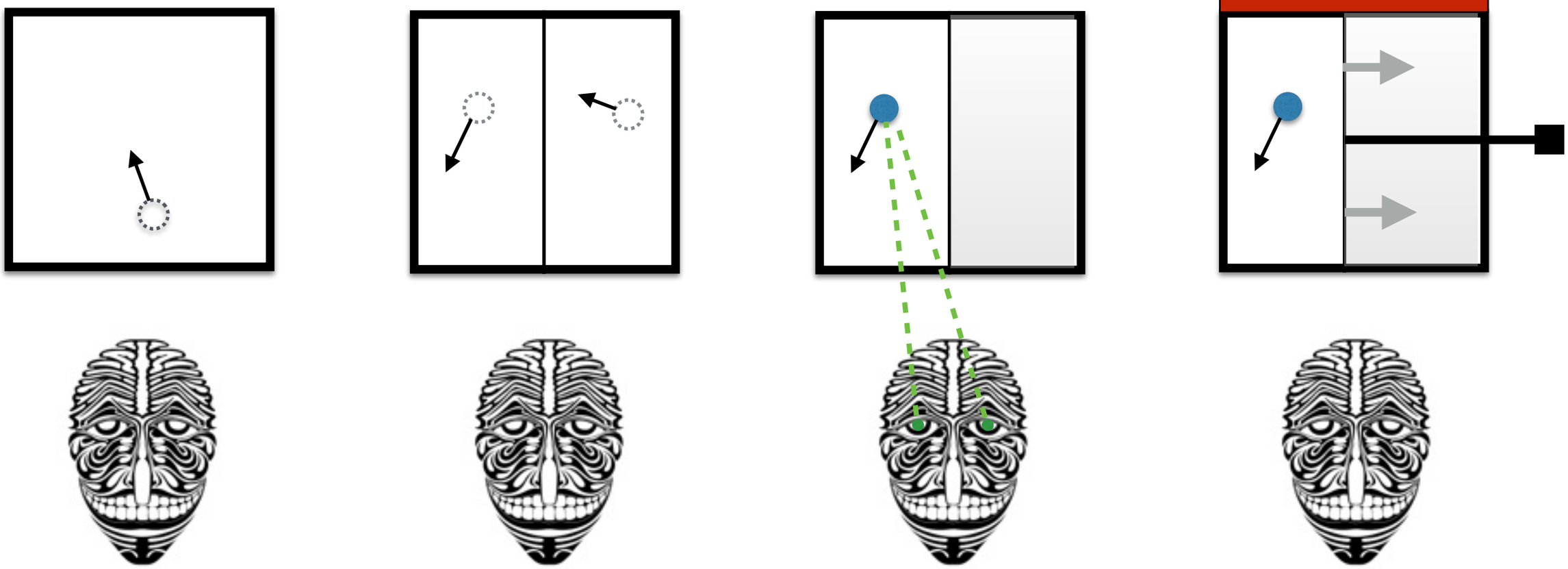
$$-T_1 \Delta S_1 = \Delta Q = T_2 \Delta S_2$$

$$\Delta S = \Delta S_2 + \Delta S_1 = \left(1 - \frac{T_2}{T_1}\right) \Delta S_2 \quad = \begin{cases} > 0 & T_1 > T_2 \\ < 0 & T_1 < T_2 \end{cases}$$

“If someone points out to you that your pet theory of the universe is in disagreement with Maxwell’s equations — then so much the worse for Maxwell’s equations. If it is found to be contradicted by observation — well these experimentalists do bungle things sometimes. But if your theory is found to be against the second law of thermodynamics I can give you no hope; there is nothing for it but to collapse in deepest humiliation.”

–Arthur Stanley Eddington

1 2 3 4



$$\Delta W = \Delta Q = -kT \ln 2 \qquad \Delta S_{bath} = -k \ln 2$$

In the cycle 1-2-3-4-1, Maxwell's demon seems to violate the second law of thermodynamics

What generates the missing entropy?

- It's the measurement! - Brillouin '51, Gabor '61.
- No, it's not the measurement! It's the change in *internal state* of the demon, viewed as changing over the course of a cyclic computational process - Landauer '61, Fredkin & Toffoli '82, Bennett '82.

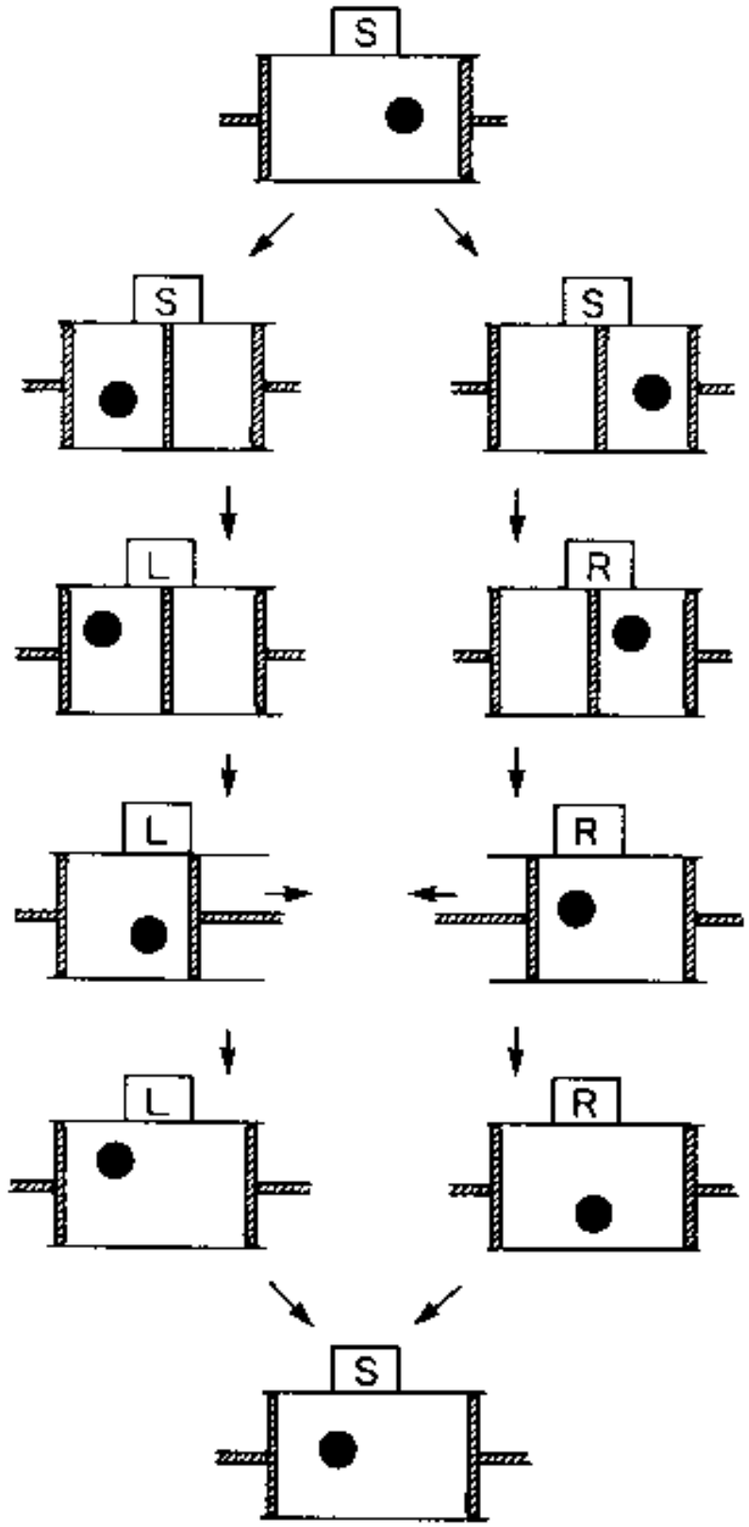
Step 1

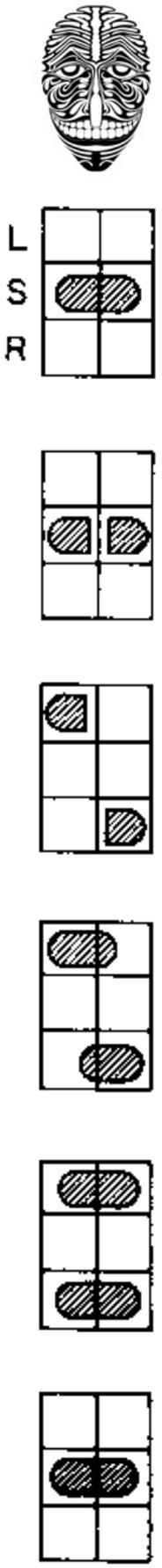
Step 2

Step 3

Step 4

$\Delta S = -k \ln 2$





(diagrams from Bennett ’82)

$\Delta S = k \ln 2$

erasure

- **Landauer's principle:** the only fundamental computational operation which generates entropy is *erasure*.

- **Bennett's exorcism**: the resolution of Maxwell's paradox is that, in order to complete a cycle, the demon's internal state must be restored to its original configuration by *erasing its memory* - thus generating the entropy necessary to put the engine back into compatibility with the second law.

II. Russell's paradox

How much time and space must be used
in carrying out a computation?

Comprehension considered harmful

$$R = \{x \mid x \notin x\}$$

$$R \in R \vdash R \notin R$$

$$R \notin R \vdash R \in R$$

The **sequent** $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ says:
together the A_i prove at least one of the B_i

$$\frac{}{A \vdash A}\ \text{axiom} \qquad \frac{\Gamma \vdash \Delta, A \qquad A, \Gamma' \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ \text{cut}$$

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\ \text{ctr} \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta} \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}$$

$$\frac{\Gamma \vdash B, B, \Delta}{\Gamma \vdash B, \Delta}\ \text{ctr} \qquad \frac{\Gamma \vdash B_1, B_2, \Delta}{\Gamma \vdash B_1 \vee B_2, \Delta} \qquad \frac{\Gamma, A_1, A_2 \vdash \Delta}{\Gamma, A_1 \wedge A_2 \vdash \Delta}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash B, \Delta}\ \text{weak} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, B \vdash \Delta}\ \text{weak} \qquad \bullet \quad \bullet \quad \bullet$$

Γ, Δ stand for multisets of formulae

Example of a proof in the sequent calculus

$$\dfrac{\dfrac{\dfrac{}{x \vdash x}\,\mathrm{ax} \qquad \dfrac{\dfrac{}{y \vdash y}\,\mathrm{ax}}{x, y \vdash y}\,\mathrm{weak}}{x, y \vdash y}\,\mathrm{cut}}{x \wedge y \vdash y}$$

Theorem (Gentzen 1934). There is an algorithm which, given a proof, produces a cut-free proof of the same sequent. This algorithm is called *cut-elimination.*

logic	**programming**
formula	type
sequent	input/output spec
proof	program
cut-elimination	execution
contraction	copying
weakening	erasure

Curry-Howard correspondence

unrestricted comprehension

$$\frac{\Gamma \vdash \Delta, A[t/x]}{\Gamma \vdash \Delta, t \in \{x \mid A\}}\,\text{comp} \qquad \frac{\Gamma, A[t/x] \vdash \Delta}{\Gamma, t \in \{x \mid A\} \vdash \Delta}\,\text{comp}$$

$$A = x \notin x \qquad R = \{x \mid A\} = \{x \mid x \notin x\}$$

$$R \notin R = A[R/x]$$

Applying the new comprehension rule yields

$$\frac{\begin{array}{c}\vdots\\ \Gamma \vdash \Delta, R \notin R\end{array}}{\Gamma \vdash \Delta, R \in R}\,\text{comp} \qquad \frac{\begin{array}{c}\vdots\\ \Gamma, R \notin R \vdash \Delta\end{array}}{\Gamma, R \in R \vdash \Delta}\,\text{comp}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{R \in R \vdash R \in R}\,\text{ax}}{\vdash R \notin R, R \in R}}{\vdash R \in R, R \in R}\,\text{comp} \qquad \dfrac{\dfrac{\dfrac{}{R \notin R \vdash R \notin R}\,\text{ax}}{R \notin R, R \in R \vdash}}{R \in R, R \in R \vdash}\,\text{comp}}{\vdash R \in R}\,\text{ctr} \qquad \dfrac{}{R \in R \vdash}\,\text{ctr}}{\vdash}\,\text{cut}$$

$$\dfrac{\vdash}{\vdash C}\,\text{weak}$$

- The sequent calculus of naive set theory is inconsistent.
- The standard point of view is that the culprit is unrestricted comprehension.

logic	**programming**
formula	type
sequent	input/output spec
proof	program
cut-elimination	execution
contraction	copying
weakening	erasure

Curry-Howard correspondence

Theorem (Girard ‘98): there is a consistent refinement of the sequent calculus with unrestricted comprehension but *restricted contraction* in which the provably total functions $\mathbb{N} \longrightarrow \mathbb{N}$ are precisely the polynomial time functions.

- This refinement is called *light linear logic.*
- The subject of *implicit computational complexity* views time and space complexity of programs as constraints on the static “geometry” of proofs.
- This has been used to propose versions of System F (the programming language on which Haskell is based) with polytime as a typing constraint (Atassi-Baillot-Terui ’07).
- Derived from a close analysis of Russell’s paradox.

logic	programming
formula	type
proof	program
cut-elimination	execution
contraction	copying
weakening	erasure
restricted contraction	restricted complexity

Russell's paradox

Maxwell's paradox

What is the physical meaning of complexity?

Simple ideal gas

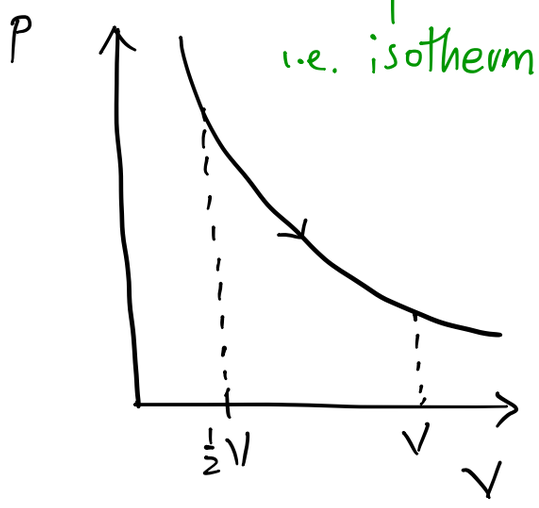
$$PV = NRT$$

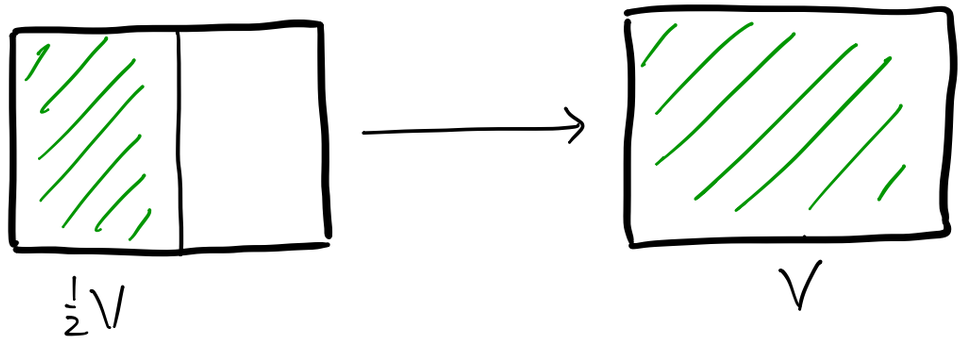
- P: pressure
- V: volume
- N: mole number
- R: universal gas constant
- T: temperature

$$U = cNRT$$

- U: energy
- c: constant, depends on gas

At fixed N, T (i.e. isothermal)





$$W = \int_{\frac{1}{2}V}^{V} P\,dV = \int_{\frac{1}{2}V}^{V} \frac{NRT}{V}\,dV$$

$$= NRT\int_{\frac{1}{2}V}^{V} \frac{1}{V}\,dV = NRT\ln\left(\frac{V}{\frac{1}{2}V}\right)$$

$$= NRT\ln 2$$

$\therefore$ Work per mole $W/N = RT\ln 2$.