What is a Resource?

Ed Blakey
edward.blakey@queens.ox.ac.uk

Complexity Resources in Physical Computation
25.viii.2009

Oxford University Computing Laboratory

The problem: factorization.

Given a natural number n, what are the prime factors of n?

The problem: factorization.

Given a natural number *n*, what are the prime factors of *n*?

We wish to design a *system* (an algorithm, Turing machine, analogue computer, quantum computer, DNA computer, …) that solves this problem using only reasonable *resource* (time, space, energy, etc.).

The problem: factorization.

Given a natural number n, what are the prime factors of n?

We wish to design a *system* (an algorithm, Turing machine, analogue computer, quantum computer, DNA computer, …) that solves this problem using only reasonable *resource* (time, space, energy, etc.).

The problem: factorization.

Given a natural number *n*, what are the prime factors of *n*?

We wish to design a *system* (an algorithm, Turing machine, analogue computer, quantum computer, DNA computer, …) that solves this problem using only reasonable *resource* (time, space, energy, etc.).

Progress so far.

Despite the attempts of countless mathematicians, computer scientists, etc., **not much progress!**

- Classical, *algorithmic* solutions (Turing machines, random-access machines, etc.) have *exponential run-time*.

Progress so far.

Despite the attempts of countless mathematicians, computer scientists, etc., **not much progress!**

- Classical, *algorithmic* solutions (Turing machines, random-access machines, etc.) have *exponential run-time*.
- *Quantum-computer* solutions (notably Shor's algorithm) are *technologically impracticable* for all but a handful of small *n*.

Progress so far.

Despite the attempts of countless mathematicians, computer scientists, etc., **not much progress!**

- Classical, *algorithmic* solutions (Turing machines, random-access machines, etc.) have *exponential run-time*.
- *Quantum-computer* solutions (notably Shor's algorithm) are *technologically impracticable* for all but a handful of small n.

So, is factorization difficult?

Not necessarily: we have neither efficient solution nor *proof* that there is no efficient solution (e.g. proof that factorization is NP-hard).

Progress so far.

Despite the attempts of countless mathematicians, computer scientists, etc., **not much progress!**

- Classical, *algorithmic* solutions (Turing machines, random-access machines, etc.) have *exponential run-time*.
- *Quantum-computer* solutions (notably Shor's algorithm) are *technologically impracticable* for all but a handful of small n.

So, is factorization difficult?

Not necessarily: we have neither efficient solution nor *proof* that there is no efficient solution (e.g. proof that factorization is NP-hard).

So, why not try other models of computation, such as *analogue computers*?

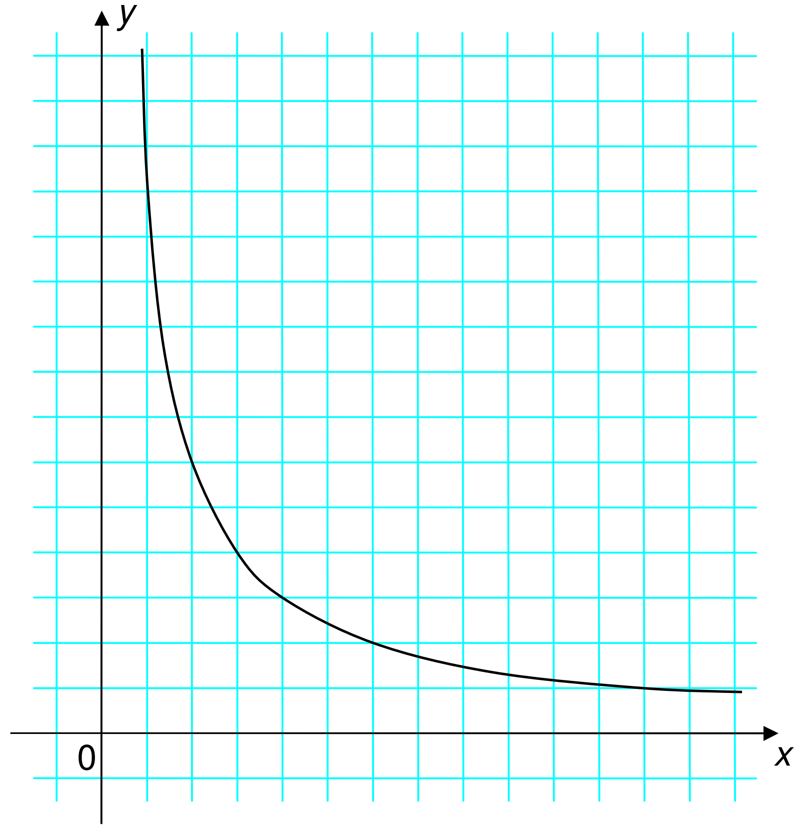
Geometric formulation.

Descartes tells us that *numerical* problems can often be recast *geometrically*.

So, don't think about finding **numbers** x and y such that $xy = n$ (i.e. $y = n/x$);
think about the **graph** $y = n/x$.

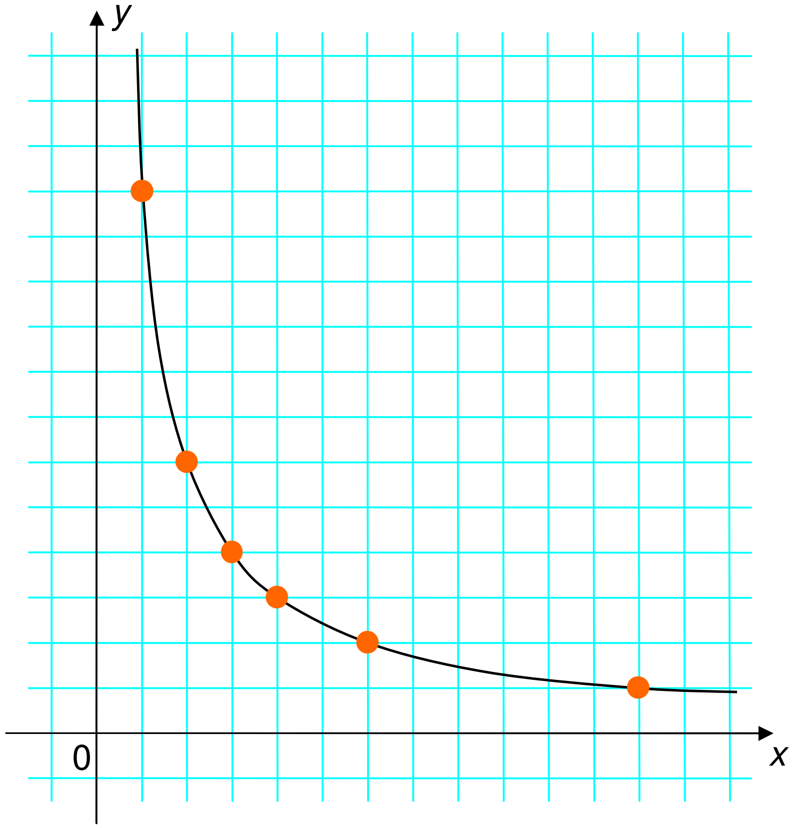
Analogue factorization system.

We want to factorize n. So, we want to find *integer* points (x, y) on the curve $y = n/x$. Such x and y are *factors of n*.



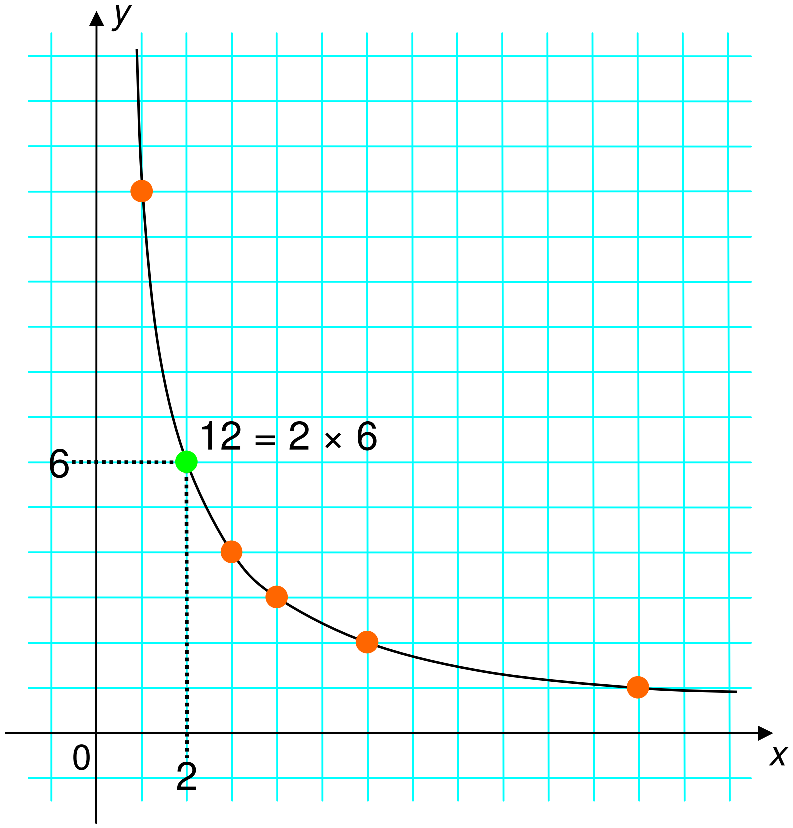
Analogue factorization system.

We want to factorize n. So, we want to find *integer* points (x, y) on the curve $y = n/x$. Such x and y are *factors of n*.



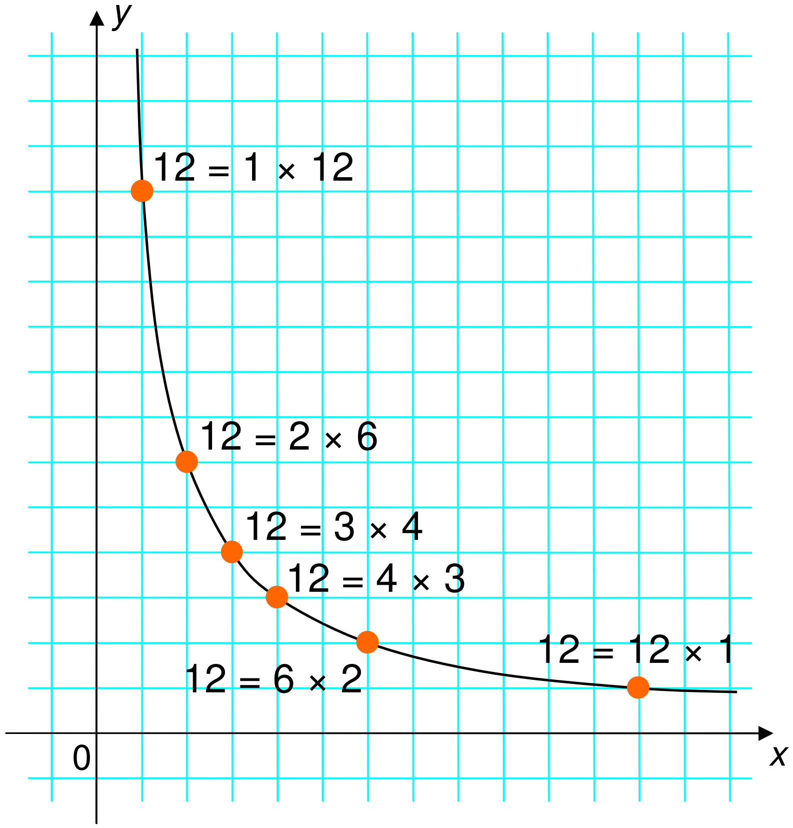
Analogue factorization system.

We want to factorize n. So, we want to find *integer* points (x, y) on the curve $y = n/x$. Such x and y are *factors of n*.



Analogue factorization system.

We want to factorize n. So, we want to find *integer* points (x, y) on the curve $y = n/x$. Such x and y are *factors of n*.



Analogue factorization system.

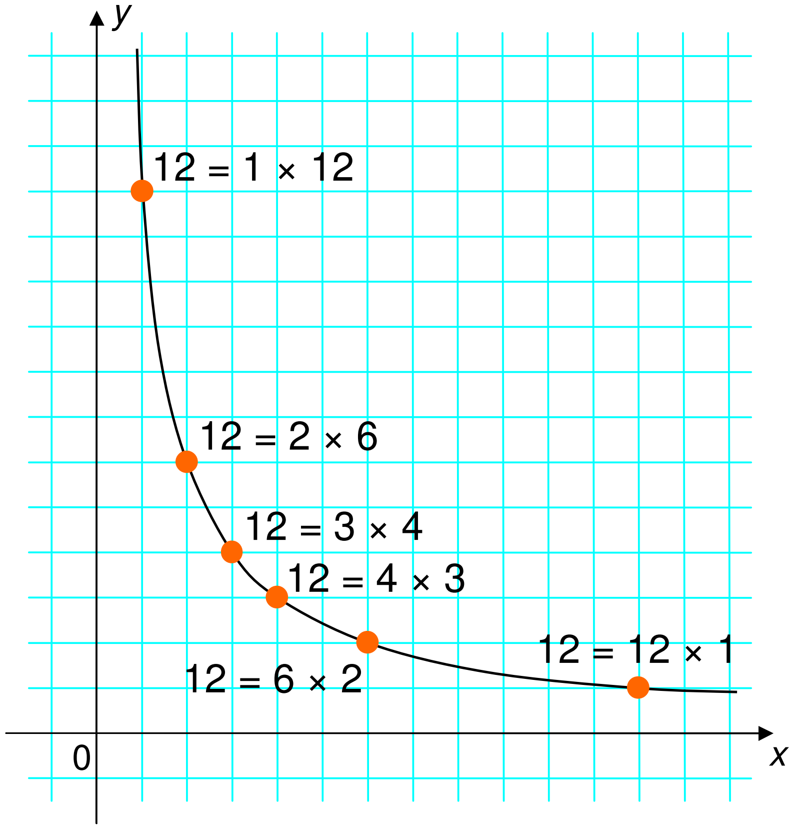
We want to factorize n. So, we want to find *integer* points (x, y) on the curve $y = n/x$. Such x and y are *factors of n*.

The curve is a hyperbola, and, hence, a *conic section*.

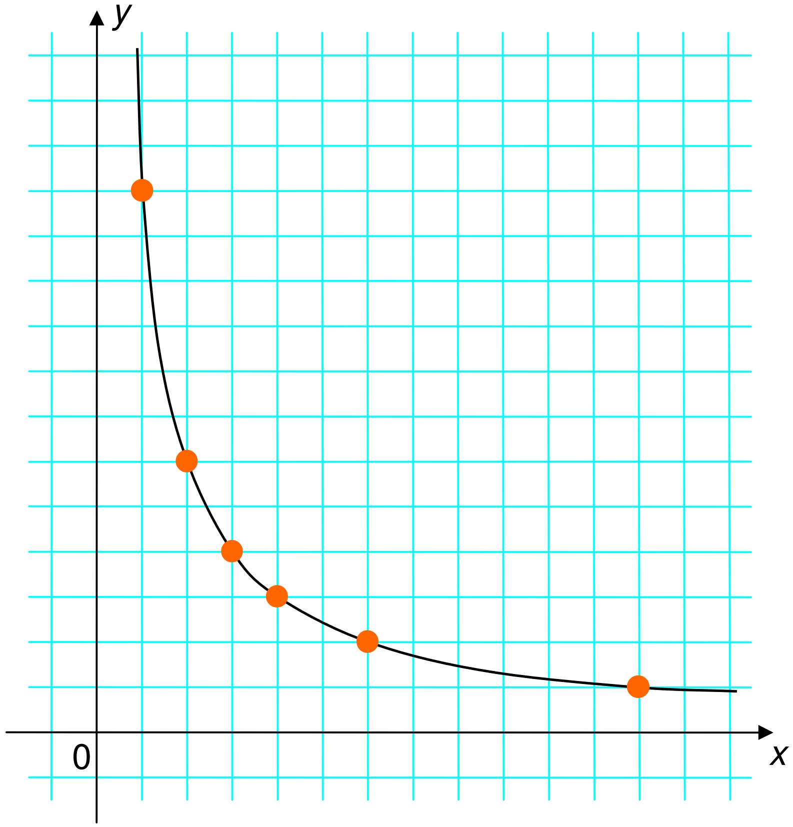
So, ***we seek points that are both***

- ***on a* cone *and***
- ***on the* integer grid**.

We implement this cone and grid.

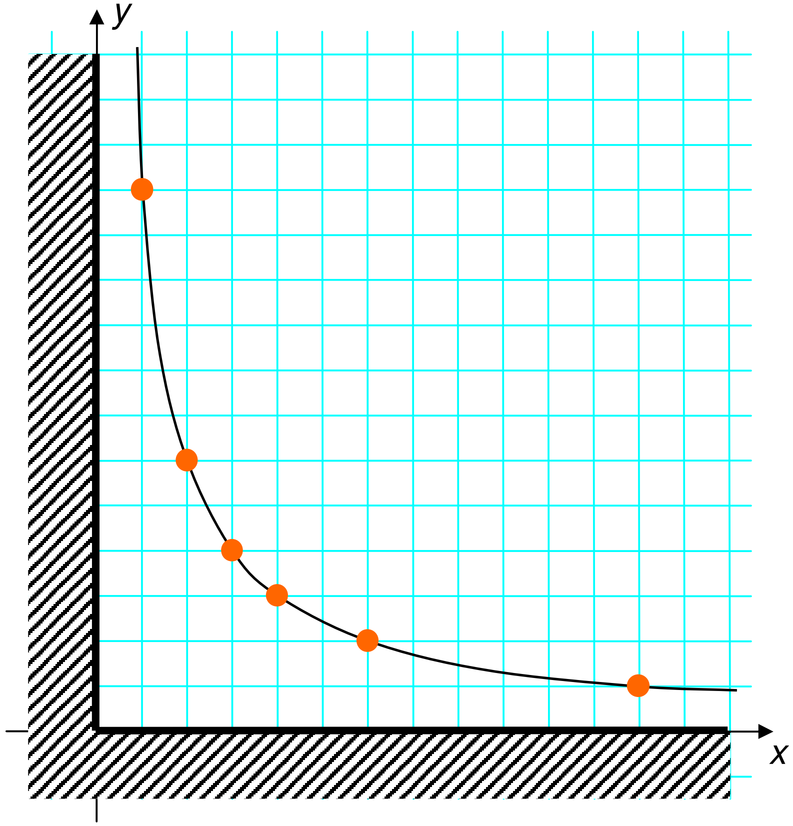


Implementing the grid.



We need only a *finite subset* of the grid

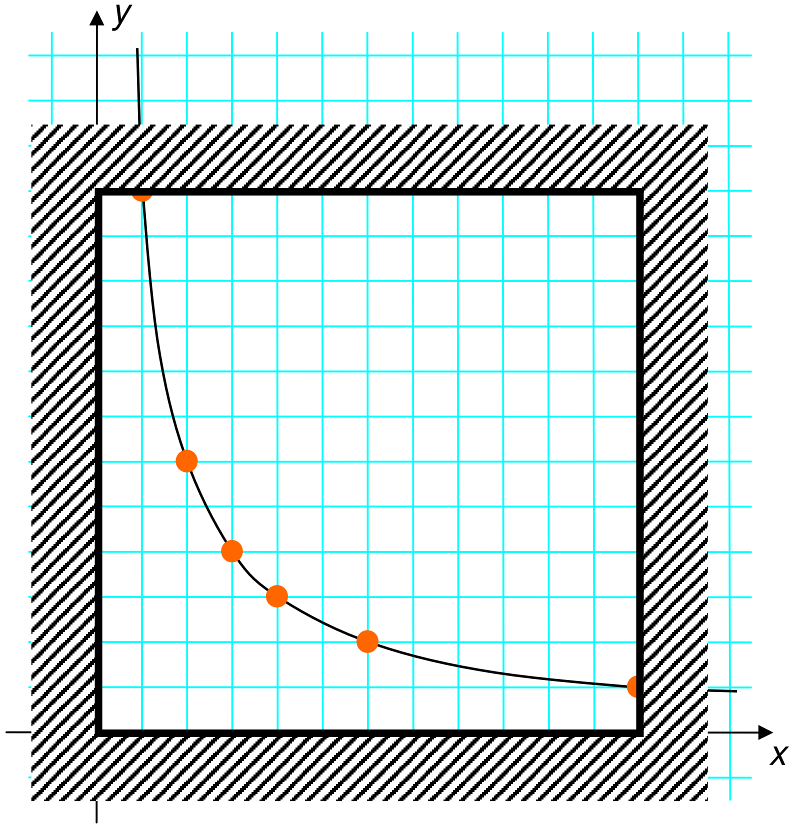
Implementing the grid.



We need only a *finite subset* of the grid:

- Factors are ≥ 0 (in fact, ≥ 1), so $\boldsymbol{x, y \geq 0}$.

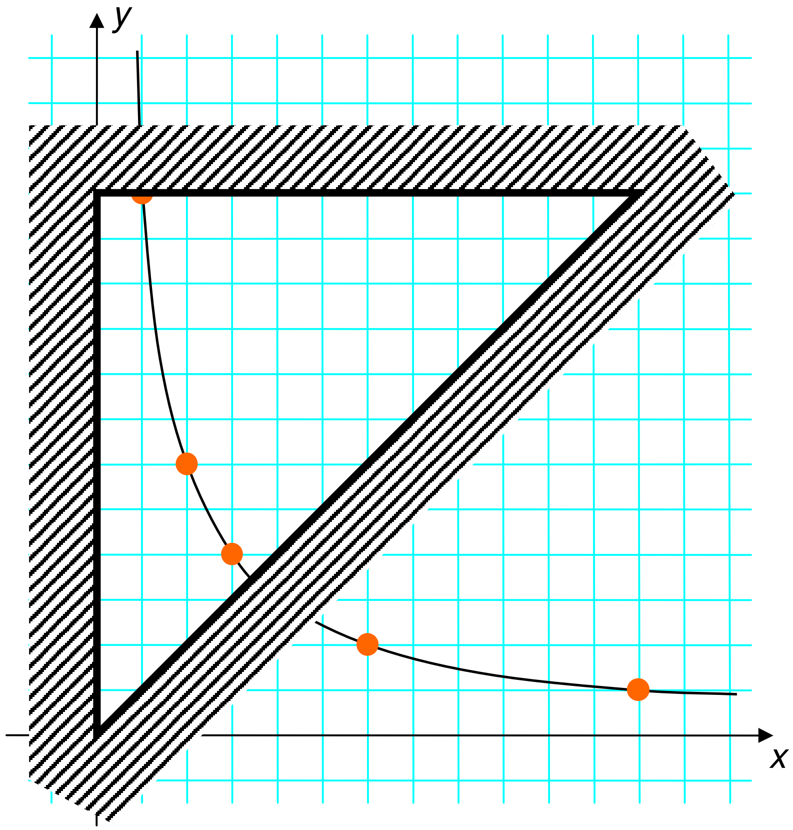
Implementing the grid.



We need only a *finite subset* of the grid:

- Factors are ≥ 0 (in fact, ≥ 1), so ***x, y* ≥ 0**.
- Factors are ≤ *n*, so ***x, y* ≤ *n***.

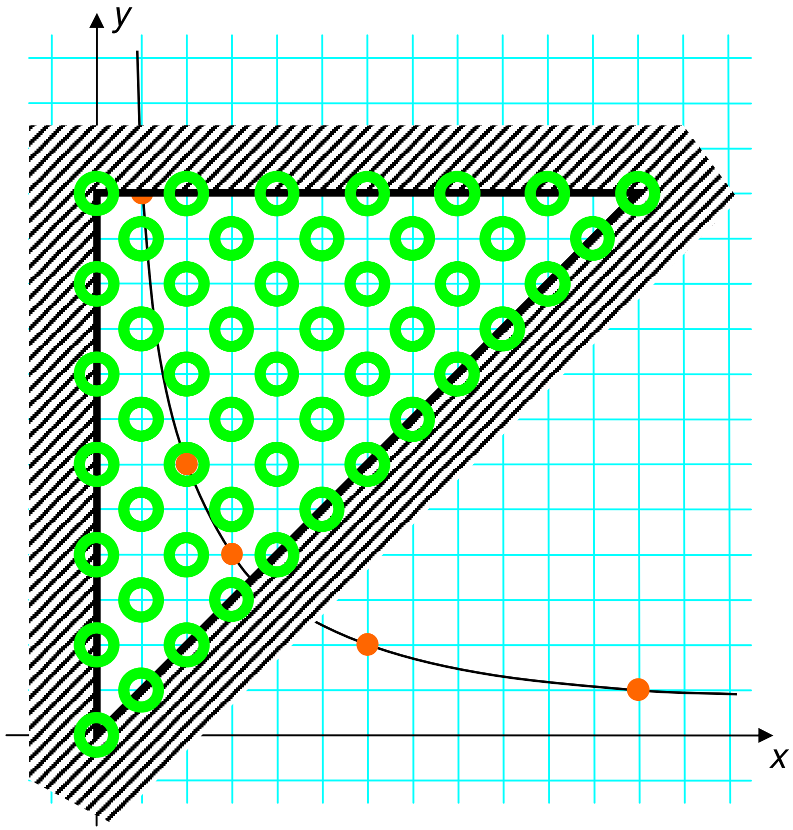
Implementing the grid.



We need only a *finite subset* of the grid:

- Factors are ≥ 0 (in fact, ≥ 1), so ***x*, *y* ≥ 0**.
- Factors are ≤ *n*, so ***x*, *y* ≤ *n***.
- (*x*, *y*) gives the same factors as (*y*, *x*), so we suppose that ***x* ≤ *y***.

Implementing the grid.

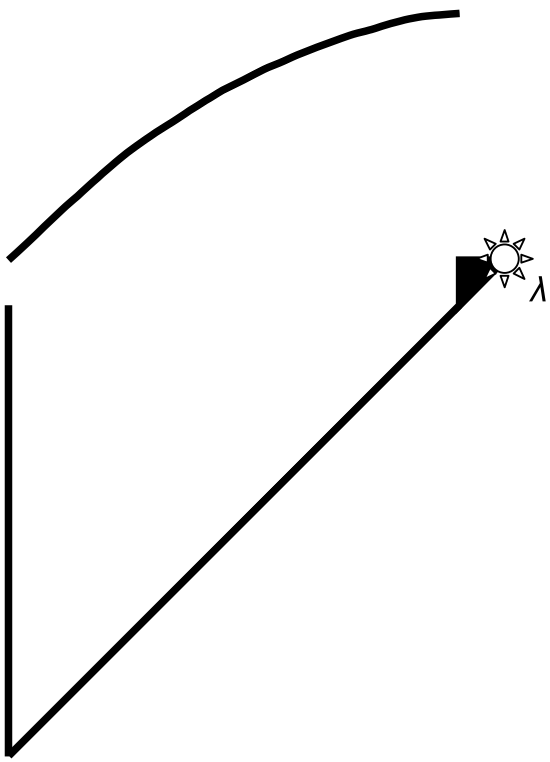


We need only a *finite subset* of the grid:

- Factors are ≥ 0 (in fact, ≥ 1), so ***x*, *y* ≥ 0**.
- Factors are ≤ *n*, so ***x*, *y* ≤ *n***.
- (*x*, *y*) gives the same factors as (*y*, *x*), so we suppose that ***x* ≤ *y***.
- We assume that *n* is odd, and so need only consider points where *x* and *y* are odd; we implement points where ***x* and *y* have the same parity**.

This is the part of the integer grid that we implement.

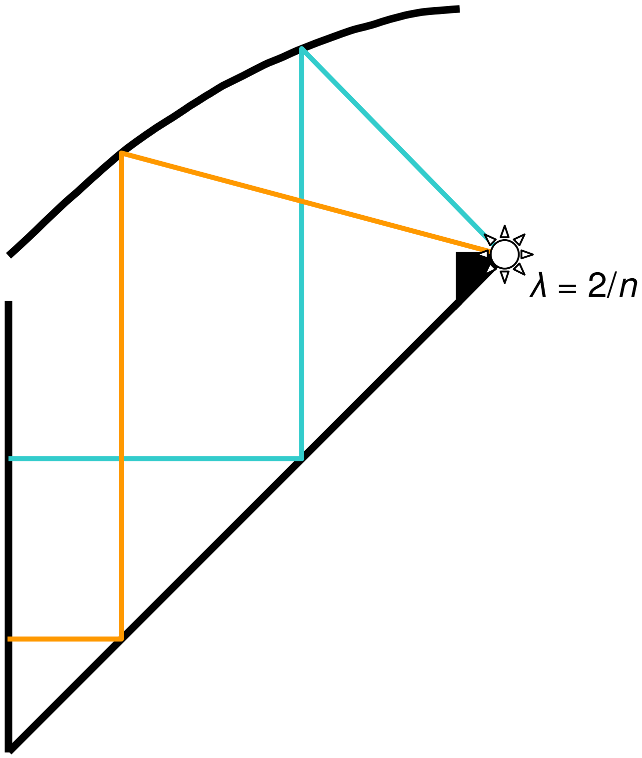
Implementing the grid.



We use:

- a *source of waves* of wavelength $\lambda = 2/n$, and
- three *mirrors* (one parabolic, two plane).

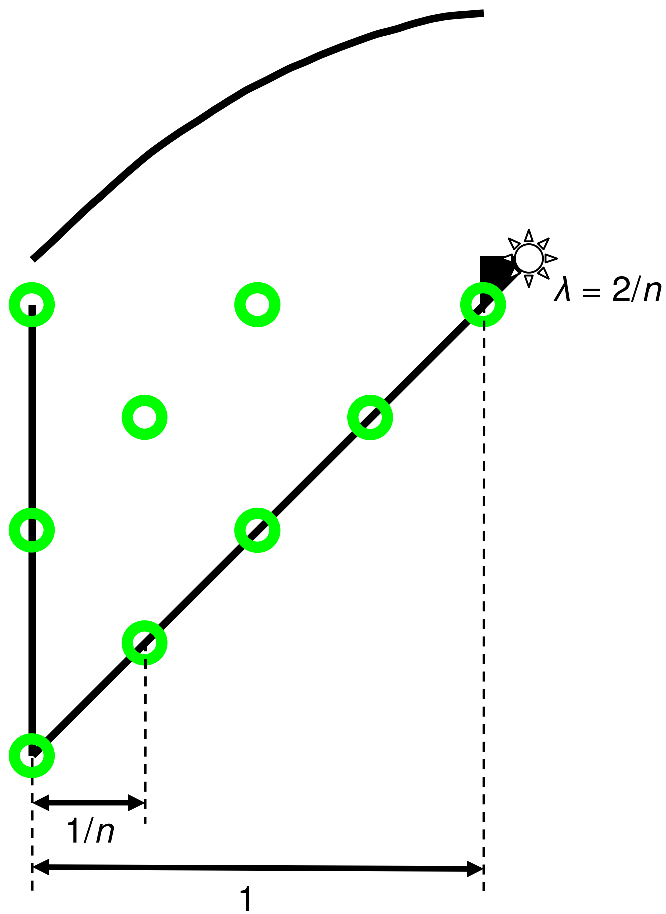
Implementing the grid.



We use:

- a *source of waves* of wavelength $\lambda = 2/n$, and
- three *mirrors* (one parabolic, two plane).

Implementing the grid.



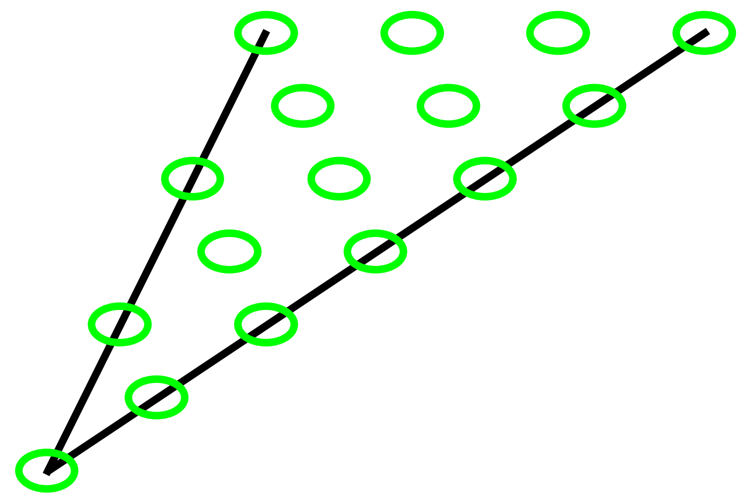
We use:

- a *source of waves* of wavelength $\lambda = 2/n$, and
- three *mirrors* (one parabolic, two plane).

The points of maximal wave activity in the resultant interference pattern model the grid points.

Since λ depends on the input value n, setting the wavelength forms part of the system's input process.

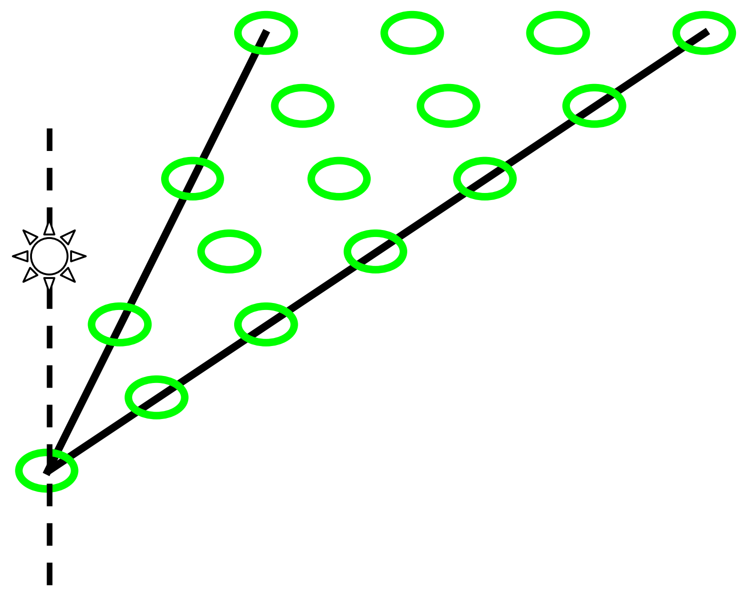
Implementing the cone.



Implementing the cone.

We use:

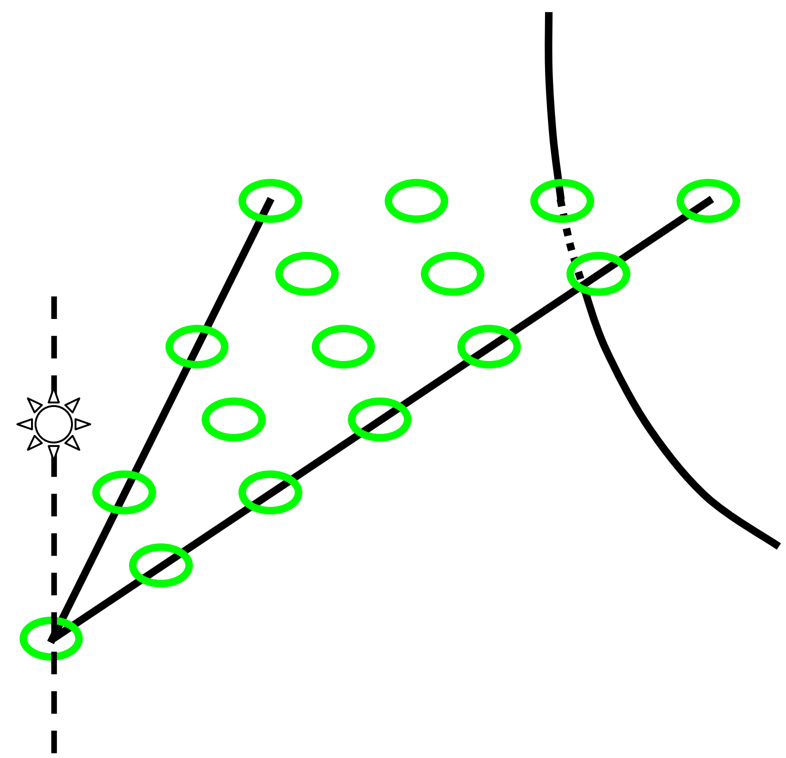
- a second *source of waves*—this is the vertex of the cone



Implementing the cone.

We use:

- a second *source of waves*—this is the vertex of the cone—, and
- a *sensor* occupying a circular arc—the circle is a cross section of the cone.

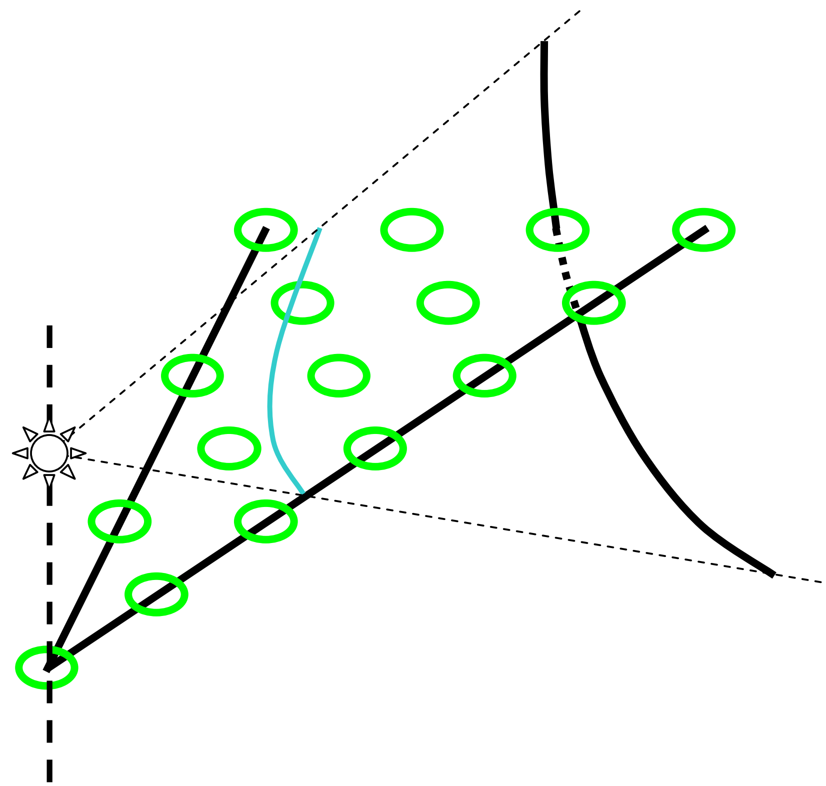


Implementing the cone.

We use:

- a second *source of waves*—this is the vertex of the cone—, and
- a *sensor* occupying a circular arc—the circle is a cross section of the cone.

Radiation from the second source arriving at the sensor passes through the grid's plane at a point on our hyperbola.



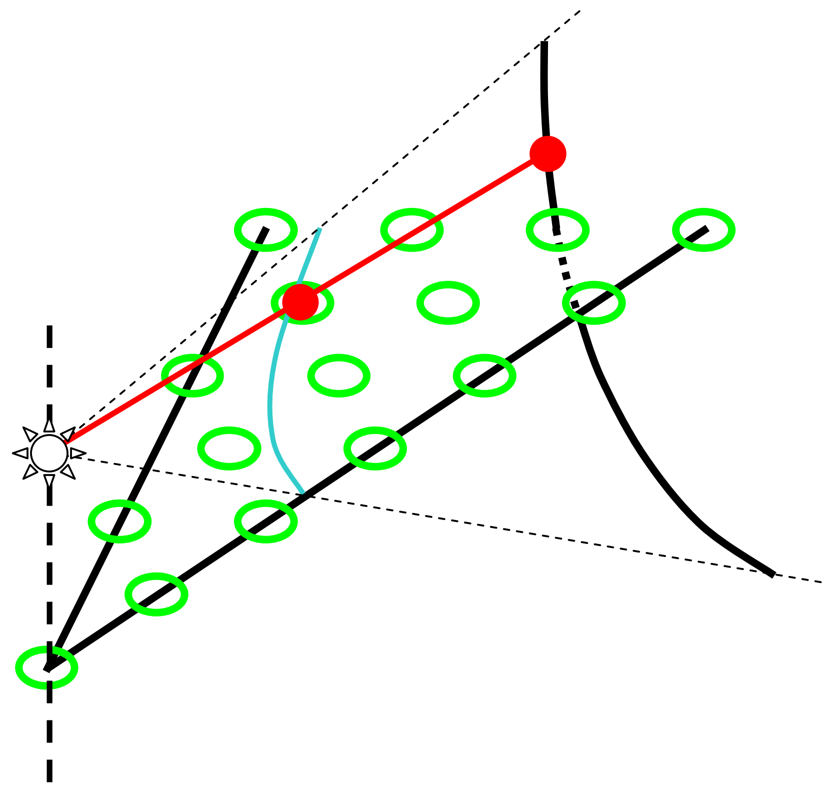
Implementing the cone.

We use:

- a second *source of waves*—this is the vertex of the cone—, and
- a *sensor* occupying a circular arc—the circle is a cross section of the cone.

Radiation from the second source arriving at the sensor passes through the grid's plane at a point on our hyperbola.

If this point is also an integer point, then the radiation will appear diminished at the sensor.



Finding factors.

- **Input.** Set parameters that depend on n: wavelength of first source, height of second source, height of sensor.
- **Processing.** Waves propagate and produce interference pattern (esp. on sensor).
- **Output.** Measure positions of ‘dark spots’ on sensor; convert these into positions of sought (grid/cone) points $\Rightarrow$ factors of n.

Finding factors.

- **Input.** Set parameters that depend on n: wavelength of first source, height of second source, height of sensor.
- **Processing.** Waves propagate and produce interference pattern (esp. on sensor).
- **Output.** Measure positions of ‘dark spots’ on sensor; convert these into positions of sought (grid/cone) points $\Rightarrow$ factors of n.

Time/space complexity.

- **Input.** Given n, calculate values (e.g. $\lambda = 2/n$) of input parameters.
- **Output.** Convert coordinates of dark spots, which encode factors of n, into factors.

These steps take *polynomial* time and space (via Turing machine).

Everything else is* constant *time/space!

Too good to be true?

Factorization in *polynomial* time and space…

What's the catch?

Too good to be true?

Factorization in *polynomial* time and space…

What's the catch?

Time and space aren't the only types of resource. The system uses an *exponential* amount of another resource, not considered in traditional complexity theory, namely…

Too good to be true?

Factorization in *polynomial* time and space…

What's the catch?

Time and space aren't the only types of resource. The system uses an *exponential* amount of another resource, not considered in traditional complexity theory, namely…

Precision.

Precision complexity captures robustness against I/O imprecision. Our system's precision complexity is *exponential*.

That is, as *n* increases, the precision with which parameters must be manipulated and measured increases *exponentially*.

How should we measure non-Turing computers' complexity?

Simply by considering all relevant resources (not just the Turing-type ones).

Given a computer, the key question is:

what resources *does the computation consume,*
and in ***what quantities****?*

Obviously still consider *algorithmic* measures (time, space, etc.) but,

especially if these seem too good to be true,

consider whether anything else *non-algorithmic (*precision, energy, etc.) is being consumed.

Types of resource.

So Turing-type resources—**time** and **space**—aren't the whole story.

In the context of unconventional computing, there's also:

Types of resource.

So Turing-type resources—**time** and **space**—aren't the whole story.

In the context of unconventional computing, there's also:

- **Precision** (e.g. in analogue, quantum, chemical and optical computers).
- **Thermodynamic cost** (in irreversible computations, where entropy increases).
- **Energy** (e.g. in mechanical/analogue and quantum-adiabatic computers).
- **Resolution** (e.g. in optical computers).
- **Weight** (e.g. in chemical computers).
- **Etc.**

'commodity'

Types of resource.

So Turing-type resources—**time** and **space**—aren’t the whole story.

In the context of unconventional computing, there’s also:

- **Precision** (e.g. in analogue, quantum, chemical and optical computers).
- **Thermodynamic cost** (in irreversible computations, where entropy increases).
- **Energy** (e.g. in mechanical/analogue and quantum-adiabatic computers).
- **Resolution** (e.g. in optical computers).
- **Weight** (e.g. in chemical computers).
- **Etc.**

Interpretations of ‘resource’.

- **‘Commodities’**. Time, space, precision, energy, weight…

Interpretations of ‘resource’.

- **‘Commodities’**. Time, space, precision, energy, weight…
- **Manufacturing cost**. Cluster states, evolution…

Interpretations of ‘resource’.

- ‘**Commodities**’. Time, space, precision, energy, weight…
- **Manufacturing cost**. Cluster states, evolution…
- **Features of the model**. Non-determinism, oracles…

Interpretations of ‘resource’.

- **‘Commodities’**. Time, space, precision, energy, weight…
- **Manufacturing cost**. Cluster states, evolution…
- **Features of the model**. Non-determinism, oracles…
- **Features of physics**. Entanglement, Newtonian dynamics…

Interpretations of ‘resource’.

- **‘Commodities’**. Time, space, precision, energy, weight…
- **Manufacturing cost**. Cluster states, evolution…
- **Features of the model**. Non-determinism, oracles…
- **Features of physics**. Entanglement, Newtonian dynamics…
- **Informatics**. E.g. ‘2 *cl-bit* + *e-bit* ≥ *qubit*’…

Interpretations of ‘resource’.

- ‘**Commodities**’. Time, space, precision, energy, weight…
- **Manufacturing cost**. Cluster states, evolution…
- **Features of the model**. Non-determinism, oracles…
- **Features of physics**. Entanglement, Newtonian dynamics…
- **Informatics**. E.g. ‘2 *cl-bit* + *e-bit* ≥ *qubit*’…

Interpretations of ‘resource’.

- **‘Commodities’**. Time, space, precision, energy, weight… Number of random bits
- **Manufacturing cost**. Cluster states, evolution…
- **Features of the model**. Non-determinism, oracles…
- **Features of physics**. Entanglement, Newtonian dynamics…
- **Informatics**. E.g. ‘2 *cl-bit* + *e-bit* ≥ *qubit*’…

Interpretations of ‘resource’.

- **‘Commodities’**. Time, space, precision, energy, weight… Number of random bits
- **Manufacturing cost**. Cluster states, evolution…
- **Features of the model**. Non-determinism, oracles…
- **Features of physics**. Entanglement, Newtonian dynamics… Non-preordained future
- **Informatics**. E.g. ‘2 *cl-bit* + *e-bit* ≥ *qubit*’…

Commodity resources.

We’ve seen some specific examples. What can we say in generality?

Commodity resources.

We've seen some specific examples. What can we say in generality?

(*Commodity*) *resources* are functions that map a computer Φ and an input value x to the number ($\in \mathbb{N}$) of units of the resource used by Φ given x.

Commodity resources.

We've seen some specific examples. What can we say in generality?

(*Commodity*) *resources* are functions that map a computer Φ and an input value x to the number ($\in \mathbb{N}$) of units of the resource used by Φ given x.

So, if T stands for the resource of *run-time*, then $T_{\Phi}(x)$ is the number of time steps required for Φ to finish its computation on input x.

Commodity resources.

We've seen some specific examples. What can we say in generality?

(*Commodity*) *resources* are functions that map a computer Φ and an input value x to the number ($\in \mathbb{N}$) of units of the resource used by Φ given x.

So, if T stands for the resource of *run-time*, then $T_\Phi(x)$ is the number of time steps required for Φ to finish its computation on input x.

We can then define the *complexity function* corresponding to a given resource:

$$TC_\Phi(n) := \sup\left\{T_\Phi(x) : |x| = n\right\}.$$

Blum's axioms.

These are conditions that a resource may or may not satisfy.

The axioms ensure:

1. that a resource is **defined** precisely at inputs at which the computation being measured is defined, and
2. that it is a [Turing-]**decidable** problem to determine whether a given value is indeed the measure of resource corresponding to a given input.

Blum's axioms.

These are conditions that a resource may or may not satisfy.

The axioms ensure:

1. that a resource is **defined** precisely at inputs at which the computation being measured is defined, and
2. that it is a [Turing-]**decidable** problem to determine whether a given value is indeed the measure of resource corresponding to a given input.

In my work (where the resources are deterministic, even if the computers being measured aren't), the axioms should hold.

But they alone aren't enough…

Dominance.

Motivation:

many resources $\Rightarrow$ comparison difficulties.

Dominance.

Motivation:

many resources $\Rightarrow$ comparison difficulties.

Comparing the time complexity of two Turing machines (to see which is more efficient) is ***easy***: use the 'big-$\mathcal{O}$' pre-ordering.

Dominance.

Motivation:

many resources $\Rightarrow$ comparison difficulties.

Comparing the time complexity of two Turing machines (to see which is more efficient) is ***easy***: use the 'big-$\mathcal{O}$' pre-ordering.

Comparing the time, space, precision, energy… complexities of, say, an optical computer with those of a chemical computer is a ***mess***: what do we $\mathcal{O}$-compare with what?

Dominance.

Motivation:

many resources $\Rightarrow$ comparison difficulties.

Comparing the time complexity of two Turing machines (to see which is more efficient) is ***easy***: use the 'big-$\mathcal{O}$' pre-ordering.

Comparing the time, space, precision, energy… complexities of, say, an optical computer with those of a chemical computer is a ***mess***: what do we $\mathcal{O}$-compare with what?

We'd like to be able to say that resource X is '***relevant***' for the optical computer, and that Y is for the chemical computer; then we can $\mathcal{O}$-compare X and Y.

Dominance formalizes this idea of 'relevance'.

Dominance.

Resource A is *dominant* if, for all B,

$$AC_{\Phi} \in \mathscr{O}(BC_{\Phi}) \implies BC_{\Phi} \in \mathscr{O}(AC_{\Phi}) \, .$$

Dominance.

We define *dominance* relative to a set of resources .

- $\mathcal{R}$ is a set of *resources*.

Resource $A \in \mathcal{R}$ is $\mathcal{R}$-*dominant* if, for all $B \in \mathcal{R}$,

$$AC_\Phi \in \mathcal{O}(BC_\Phi) \;\Rightarrow\; BC_\Phi \in \mathcal{O}(AC_\Phi)\ .$$

Dominance.

We define *dominance* relative to a set of resources and to a computer.

- $\mathcal{R}$ is a set of *resources*.
- Φ is a *computer*.

Resource $A \in \mathcal{R}$ is $\mathcal{R}$-*dominant* for Φ if, for all $B \in \mathcal{R}$,

$$AC_{\Phi} \in \mathcal{O}(BC_{\Phi}) \implies BC_{\Phi} \in \mathcal{O}(AC_{\Phi}) \ .$$

Dominance.

We define *dominance* relative to a set of resources and to a computer.

- $\mathcal{R}$ is a set of *resources*.
- Φ is a *computer*.

Resource $A \in \mathcal{R}$ is $\mathcal{R}$-*dominant* for Φ if, for all $B \in \mathcal{R}$,

$$AC_\Phi \in \mathcal{O}(BC_\Phi) \;\Rightarrow\; BC_\Phi \in \mathcal{O}(AC_\Phi)\ .$$

That is,

$\mathcal{R}$-dominant resources are those that $\mathcal{O}$-exceed all resources with which they are $\mathcal{O}$-comparable.

$\mathcal{R}$-complexity.

(As before, $\mathcal{R}$ is a set of *resources* and Φ is a *computer*.)

The $\mathcal{R}$-*complexity* of Φ, denoted $\mathcal{B}_{\mathcal{R},\, \Phi}$, is the complexity function given by:

$$\mathcal{B}_{\mathcal{R},\, \Phi}(n) \ := \ \Sigma_{A \text{ is } \mathcal{R}\text{-dominant}} \, AC_{\Phi}(n) \ .$$

We sum 'relevant' resources (and no others).

This captures 'overall complexity'.

Why Blum's axioms aren't enough.

Let

- $S_{\Phi}(x)$ be the number of tape-cells used, and
- $T_{\Phi}(x)$ the number of time steps elapsed,

during a computation by Turing machine Φ with input value x.

Why Blum's axioms aren't enough.

Let

- $S_\Phi(x)$ be the number of tape-cells used, and
- $T_\Phi(x)$ the number of time steps elapsed,

during a computation by Turing machine Φ with input value x.

Now define resource S' by $S'(x) := 2^{S(x)}$.

Why Blum's axioms aren't enough.

Let

- $S_\Phi(x)$ be the number of tape-cells used, and
- $T_\Phi(x)$ the number of time steps elapsed,

during a computation by Turing machine Φ with input value x.

Now define resource S' by $S'(x) := 2^{S(x)}$.

- As far as Blum's axioms are concerned, S and S' are **legitimate resources**.
- Each is an **internally consistent** measure of space usage.
- $S(x) \mapsto S'(x)$ is an **isotone** mapping: ordering inputs by their values of S has the same result as ordering by values of S'.

So, we have two seemingly viable, isomorphic ways of quantifying space usage.

Why Blum's axioms aren't enough.

Let

- $S_{\Phi}(x)$ be the number of tape-cells used, and
- $T_{\Phi}(x)$ the number of time steps elapsed,

during a computation by Turing machine Φ with input value x.

Now define resource S' by $S'(x) := 2^{S(x)}$.

- As far as Blum's axioms are concerned, S and S' are **legitimate resources**.
- Each is an **internally consistent** measure of space usage.
- $S(x) \mapsto S'(x)$ is an **isotone** mapping: ordering inputs by their values of S has the same result as ordering by values of S'.

So, we have two seemingly viable, isomorphic ways of quantifying space usage.

But then there's dominance…

Space vs. time.

According to our notion of dominance, which resource is more relevant, *space* or *time*?

Space vs. time.

According to our notion of dominance, which resource is more relevant, *space* or *time*?

Well, suppose that $T(x) \in \mathcal{O}(n^2)$ and $S(x) \in \mathcal{O}(n)$ (whence $S'(x) \in 2^{\mathcal{O}(n)}$). Then:

- ***S'* dominates *T*** (i.e. *S'* is {*S'*, *T*}-dominant but *T* is not), but
- ***T* dominates *S*** (i.e. *T* is {*S*, *T*}-dominant but *S* is not).

Space vs. time.

According to our notion of dominance, which resource is more relevant, *space* or *time*?

Well, suppose that $T(x) \in \mathcal{O}(n^2)$ and $S(x) \in \mathcal{O}(n)$ (whence $S'(x) \in 2^{\mathcal{O}(n)}$). Then:

- ***S'* dominates *T*** (i.e. S' is $\{S', T\}$-dominant but T is not), but
- ***T* dominates *S*** (i.e. T is $\{S, T\}$-dominant but S is not).

Space, depending on how we measure it, can be either *more* or *less* relevant than time!

We can engineer which resource appears more important. By applying to the more slowly growing, non-dominant resource a sufficiently fast-growing, monotonic function (e.g. $n \mapsto 2^n$), this resource becomes dominant. **We don't want this!**

Space vs. time.

According to our notion of dominance, which resource is more relevant, *space* or *time*?

Well, suppose that $T(x) \in \mathcal{O}(n^2)$ and $S(x) \in \mathcal{O}(n)$ (whence $S'(x) \in 2^{\mathcal{O}(n)}$). Then:

- ***S'* dominates *T*** (i.e. S' is $\{S', T\}$-dominant but T is not), but
- ***T* dominates *S*** (i.e. T is $\{S, T\}$-dominant but S is not).

Space, depending on how we measure it, can be either *more* or *less* relevant than time!

We can engineer which resource appears more important. By applying to the more slowly growing, non-dominant resource a sufficiently fast-growing, monotonic function (e.g. $n \mapsto 2^n$), this resource becomes dominant. **We don't want this!**

So, let's restrict resources (so that, e.g., S is valid but S' is not) to stop this sort of thing. We do this with ***normalization***.

Normalization—motivation.

Recall resources S (the number of tape-cells) and S' (= 2^S).

What values can these resources take?

- We can write to any number of cells, then halt. So S maps surjectively to $\mathbb{N}$.
- Hence, S' maps surjectively to $\{1, 2, 4, 8, \ldots\}$.

The former property—mapping onto $\mathbb{N}$—seems the more natural (quite literally!), and we use it as our blueprint for normalization.

Normalization—definition.

Let $\mathcal{C}$ be a class of computers. For each $\Phi \in \mathcal{C}$, let X_Φ be the set of input values for Φ.

Normalization—definition.

Let $\mathcal{C}$ be a class of computers. For each $\Phi \in \mathcal{C}$, let X_Φ be the set of input values for Φ.

- Let A be a resource that can take as its subscript any computing system $\Phi \in \mathcal{C}$ (A_Φ: $X_\Phi \to \mathbb{N}$). Define the $\mathcal{C}$-*normalized form* of A to be the resource $A^\mathcal{C}$ given by $A^\mathcal{C}{}_\Phi$: $X_\Phi \to \mathbb{N}$,

$$A^\mathcal{C}{}_\Phi(x) := |\{\, A_\Psi(y) : \Psi \in \mathcal{C} \text{ and } y \in X_\Psi \text{ and } A_\Psi(y) < A_\Phi(x)\}|\ .$$

Normalization—definition.

Let $\mathcal{C}$ be a class of computers. For each $\Phi \in \mathcal{C}$, let X_Φ be the set of input values for Φ.

- Let A be a resource that can take as its subscript any computing system $\Phi \in \mathcal{C}$ (A_Φ: $X_\Phi \to \mathbb{N}$). Define the $\mathcal{C}$-*normalized form* of A to be the resource $A^\mathcal{C}$ given by $A^\mathcal{C}{}_\Phi$: $X_\Phi \to \mathbb{N}$,

$$A^\mathcal{C}{}_\Phi(x) := |\{ A_\Psi(y) : \Psi \in \mathcal{C} \text{ and } y \in X_\Psi \text{ and } A_\Psi(y) < A_\Phi(x)\}| \ .$$

- Resource A is $\mathcal{C}$-*normal* if $A = A^\mathcal{C}$ (i.e. if $A_\Phi(x) = A^\mathcal{C}{}_\Phi(x)$ for all $\Phi \in \mathcal{C}$ and $x \in X_\Phi$).

Normalization—definition.

Let $\mathcal{C}$ be a class of computers. For each $\Phi \in \mathcal{C}$, let X_Φ be the set of input values for Φ.

- Let A be a resource that can take as its subscript any computing system $\Phi \in \mathcal{C}$ ($A_\Phi\colon X_\Phi \to \mathbb{N}$). Define the $\mathcal{C}$-*normalized form* of A to be the resource $A^\mathcal{C}$ given by $A^\mathcal{C}{}_\Phi\colon X_\Phi \to \mathbb{N}$,

$$A^\mathcal{C}{}_\Phi(x) := |\{\, A_\Psi(y) : \Psi \in \mathcal{C} \text{ and } y \in X_\Psi \text{ and } A_\Psi(y) < A_\Phi(x)\}|\ .$$

- Resource A is $\mathcal{C}$-*normal* if $A = A^\mathcal{C}$ (i.e. if $A_\Phi(x) = A^\mathcal{C}{}_\Phi(x)$ for all $\Phi \in \mathcal{C}$ and $x \in X_\Phi$).

In words, $A^\mathcal{C}{}_\Phi(x)$ is the number of distinct values less than $A_\Phi(x)$ taken by A (as it ranges over all computers in $\mathcal{C}$ and all input values).

This is a measure of ‘how much use A makes’ of the natural numbers less than $A_\Phi(x)$.

Normalization—example.

Revisiting our example of S (the number of tape-cells) and S' (= 2^S), we have that

$$S'^{\mathfrak{T}} = S^{\mathfrak{T}} = S$$

(and that $T^{\mathfrak{T}} = T$),

where $\mathfrak{T}$ is the class of Turing machines.

So, if we use only *normal* resources, we may validly compare S and T, but not S' and T.

Normalization—properties.

- Normalization is strictly **isotone**:

$$A_{\Phi}(x) < A_{\Psi}(y) \text{ if and only if } A^{\mathcal{C}}{}_{\Phi}(x) < A^{\mathcal{C}}{}_{\Psi}(y)\ .$$

Normalization—properties.

- Normalization is strictly **isotone**:

$$A_{\Phi}(x) < A_{\Psi}(y) \text{ if and only if } A^{\mathcal{c}}{}_{\Phi}(x) < A^{\mathcal{c}}{}_{\Psi}(y) \ .$$

- Normalization is **idempotent**:

$$A^{\mathcal{c}\mathcal{c}}{}_{\Phi}(x) = A^{\mathcal{c}}{}_{\Phi}(x) \ .$$

Normalization—properties.

- Normalization is strictly **isotone**:

$$A_{\Phi}(x) < A_{\Psi}(y) \text{ if and only if } A^{\mathfrak{C}}{}_{\Phi}(x) < A^{\mathfrak{C}}{}_{\Psi}(y)\ .$$

- Normalization is **idempotent**:

$$A^{\mathfrak{C}\mathfrak{C}}{}_{\Phi}(x) = A^{\mathfrak{C}}{}_{\Phi}(x)\ .$$

- **Characterization** of normal resources:

Resource A is normal if and only if its image set (over all computers and input values) $\{A_{\Psi}(y) : \Psi \in \mathfrak{C} \text{ and } y \in X_{\Psi}\}$ is an 'initial segment' $\{i \in \mathbb{N} : i < n\}$ for some $n \in \mathbb{N} \cup \{\infty\}$.

Normalization—intuition.

Non-normalized: cardinal.
Normalized: ordinal.

If we let resources be *any* functions that satisfy Blum's axioms and have codomain $\mathbb{N}$, then we are effectively dealing with ***cardinals***: we are *counting* time steps, units of energy or similar—we have an intrinsic *unit of measurement*.

This is resource-dependent and not conducive to resource-heterogeneous comparison (e.g., how many time steps should we deem of equivalent cost/value to one tape cell?).

Normalization—intuition.

Non-normalized: cardinal.
Normalized: ordinal.

If we let resources be *any* functions that satisfy Blum's axioms and have codomain $\mathbb{N}$, then we are effectively dealing with ***cardinals***: we are *counting* time steps, units of energy or similar—we have an intrinsic *unit of measurement*.

This is resource-dependent and not conducive to resource-heterogeneous comparison (e.g., how many time steps should we deem of equivalent cost/value to one tape cell?).

But if we allow only $\mathcal{C}$-*normal* resources, then we have ***ordinals***: 0 represents the *least* resource consumption, 1 the *second-least*, 2 the *third-least*, and so on; this is independent of resources and units. Comparison then seems fair and equal-footed.

Concluding comments.

- With *Turing machines*, we know what resources to consider.
- With *unconventional computers*, it's not so obvious.
- And even when we've identified our unconventional resources, there are comparison difficulties. Hence ***dominance***.
- But to make dominance work, we need to restrict our notion of resource (more than Blum's axioms do). Hence ***normalization***.
- This restriction stops certain 'deceptive' complexity behaviour, e.g. 'dominance engineering' via application of quickly growing, isotone functions.
- The restriction also renders resources 'ordinal', not 'cardinal', allowing seemingly fairer comparison.

Questions?

---

edward.blakey@queens.ox.ac.uk **http://users.ox.ac.uk/~quee1871**

A paper, ***Beyond Blum: What is a Resource?***, related to parts of this talk is to appear in the *International Journal of Unconventional Computation*.

We thank the EPSRC for its generous support. This workshop and the research described in this talk are funded by EPSRC grant

Complexity and Decidability in Unconventional Computational Models.