How do we Describe the Computational Capabilities of an Arbitrary Physical System?

Richard Whyman

The University of Leeds

9th September 2017

Prior Work on Physical Computation

Physical computation has been discussed in various places before such as by:

- Robin Gandy in *Church's thesis and principles for mechanisms* (1978).
- Horsman et al. in *When does a Physical System Compute?* (2013).
- Cameron Beebe in *Model Based Computation* (2016).
- Vergis et al. in *The complexity of analog computation* (1986).
- Ed Blakey in *Unconventional complexity measures for unconventional computers* (2011).

What's Wrong with Just Using the Turing Model?

- The Turing machine model appears to describe what is computable by a mechanism, but its method of computation does not seem to faithfully describe **how** many physical computation devices actually compute.
- Quantum computers are an example of such a device, they also appear to be able to efficiently decide problems that are in $(NP \cap \text{co-}NP) \setminus P$. Thereby potentially violating Cobham's thesis.
- Hence the Turing machine model may not be sufficient for judging, in general, whether a problem can be **feasibly** decided by some arbitrary physical device.

What's Wrong with Just Using the Turing Model?

- The Turing machine model appears to describe what is computable by a mechanism, but its method of computation does not seem to faithfully describe **how** many physical computation devices actually compute.
- Quantum computers are an example of such a device, they also appear to be able to efficiently decide problems that are in $(NP \cap \text{co-}NP) \setminus P$. Thereby potentially violating Cobham's thesis.
- Hence the Turing machine model may not be sufficient for judging, in general, whether a problem can be **feasibly** decided by some arbitrary physical device.

What's Wrong with Just Using the Turing Model?

- The Turing machine model appears to describe what is computable by a mechanism, but its method of computation does not seem to faithfully describe **how** many physical computation devices actually compute.
- Quantum computers are an example of such a device, they also appear to be able to efficiently decide problems that are in $(NP \cap \text{co-}NP) \setminus P$. Thereby potentially violating Cobham's thesis.
- Hence the Turing machine model may not be sufficient for judging, in general, whether a problem can be **feasibly** decided by some arbitrary physical device.

On Computation and Logical Consequence

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
???	$\overset{?}{\Longleftrightarrow}$	The Turing machine model of computation
$\downarrow$		$\downarrow$
A non-sequential basis for complexity?	$\overset{?}{\Longleftrightarrow}$	Sequential algorithmic basis for complexity

On Computation and Logical Consequence

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
???	$\overset{?}{\Longleftrightarrow}$	The Turing machine model of computation
$\downarrow$		$\downarrow$
A non-sequential basis for complexity?	$\overset{?}{\Longleftrightarrow}$	Sequential algorithmic basis for complexity

On Computation and Logical Consequence

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
???	$\overset{?}{\Longleftrightarrow}$	The Turing machine model of computation
$\downarrow$		$\downarrow$
A non-sequential basis for complexity?	$\overset{?}{\Longleftrightarrow}$	Sequential algorithmic basis for complexity

On Computation and Logical Consequence

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
???	$\overset{?}{\Longleftrightarrow}$	The Turing machine model of computation
$\downarrow$		$\downarrow$
A non-sequential basis for complexity?	$\overset{?}{\Longleftrightarrow}$	Sequential algorithmic basis for complexity

On Computation and Logical Consequence

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
???	$\overset{?}{\Longleftrightarrow}$	The Turing machine model of computation
$\downarrow$		$\downarrow$
A non-sequential basis for complexity?	$\overset{?}{\Longleftrightarrow}$	Sequential algorithmic basis for complexity

On Computation and Logical Consequence

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
???	$\overset{?}{\Longleftrightarrow}$	The Turing machine model of computation
$\downarrow$		$\downarrow$
A non-sequential basis for complexity?	$\overset{?}{\Longleftrightarrow}$	Sequential algorithmic basis for complexity

On Computation and Logical Consequence

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
???	$\overset{?}{\Longleftrightarrow}$	The Turing machine model of computation
$\downarrow$		$\downarrow$
A non-sequential basis for complexity?	$\overset{?}{\Longleftrightarrow}$	Sequential algorithmic basis for complexity

On Computation and Logical Consequence

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
???	$\overset{?}{\Longleftrightarrow}$	The Turing machine model of computation
$\downarrow$		$\downarrow$
A non-sequential basis for complexity?	$\overset{?}{\Longleftrightarrow}$	Sequential algorithmic basis for complexity

On Computation and Logical Consequence

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
???	$\overset{?}{\Longleftrightarrow}$	The Turing machine model of computation
$\downarrow$		$\downarrow$
A non-sequential basis for complexity?	$\overset{?}{\Longleftrightarrow}$	Sequential algorithmic basis for complexity

On Computation and Logical Consequence

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
???	$\overset{?}{\Longleftrightarrow}$	The Turing machine model of computation
$\downarrow$		$\downarrow$
A non-sequential basis for complexity?	$\overset{?}{\Longleftrightarrow}$	Sequential algorithmic basis for complexity

On Computation and Logical Consequence

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
???	$\overset{?}{\Longleftrightarrow}$	The Turing machine model of computation
$\downarrow$		$\downarrow$
A non-sequential basis for complexity?	$\overset{?}{\Longleftrightarrow}$	Sequential algorithmic basis for complexity

Theory Machines and Derivation

How to Compute Semantically

- Suppose that, given a set of abstract conditions $\mathcal{T}$ and an input condition ϕ, it logically follows that statement θ must be true.
- Then if a physical system $\mathfrak{P}$ satisfies the conditions in $\mathcal{T}$ along with the input condition ϕ, the statement θ must necessarily true in $\mathfrak{P}$.
- Our view is that in such a scenario; θ can be taken to be the output computed by a physical system $\mathfrak{P}$ under input ϕ and satisfying the conditions of $\mathcal{T}$.

How to Compute Semantically

- Suppose that, given a set of abstract conditions $\mathcal{T}$ and an input condition ϕ, it logically follows that statement θ must be true.
- Then if a physical system $\mathfrak{P}$ satisfies the conditions in $\mathcal{T}$ along with the input condition ϕ, the statement θ must necessarily true in $\mathfrak{P}$.
- Our view is that in such a scenario; θ can be taken to be the output computed by a physical system $\mathfrak{P}$ under input ϕ and satisfying the conditions of $\mathcal{T}$.

How to Compute Semantically

- Suppose that, given a set of abstract conditions $\mathcal{T}$ and an input condition ϕ, it logically follows that statement θ must be true.
- Then if a physical system $\mathfrak{P}$ satisfies the conditions in $\mathcal{T}$ along with the input condition ϕ, the statement θ must necessarily true in $\mathfrak{P}$.
- Our view is that in such a scenario; θ can be taken to be the output computed by a physical system $\mathfrak{P}$ under input ϕ and satisfying the conditions of $\mathcal{T}$.

Theory Machines

Definition: Let $\mathcal{L}$ be a finite first-order language, a *theory machine* over $\mathcal{L}$ is a triple $\mathcal{M} = (\mathcal{T}, \mathcal{I}, \mathcal{O})$ where:

- $\mathcal{T}$ is a finite set of sentences of $\mathcal{L}$.
  (this provides an abstract description for the machine)
- $\mathcal{I}$ is a set of distinct atomic sentences of $\mathcal{L}$.
  (this is the set of possible inputs to the machine)
- $\mathcal{O}$ is a set of distinct atomic sentences of $\mathcal{L}$.
  (this is the set of possible outputs from the machine)

(Atomic sentences are quantifier free sentences of the form $\bigwedge_{i=1}^{N} \pm R_i(\tau_{i1}, \ldots, \tau_{ir_i})$ for relations R_i and terms τ_{ik}.)

Theory Machines

Definition: Let $\mathcal{L}$ be a finite first-order language, a *theory machine* over $\mathcal{L}$ is a triple $\mathcal{M} = (\mathcal{T}, \mathcal{I}, \mathcal{O})$ where:

- $\mathcal{T}$ is a finite set of sentences of $\mathcal{L}$.
  (this provides an abstract description for the machine)
- $\mathcal{I}$ is a set of distinct atomic sentences of $\mathcal{L}$.
  (this is the set of possible inputs to the machine)
- $\mathcal{O}$ is a set of distinct atomic sentences of $\mathcal{L}$.
  (this is the set of possible outputs from the machine)

(Atomic sentences are quantifier free sentences of the form $\bigwedge_{i=1}^{N} \pm R_i(\tau_{i1}, \ldots, \tau_{ir_i})$ for relations R_i and terms τ_{ik}.)

Theory Machines

Definition: Let $\mathcal{L}$ be a finite first-order language, a *theory machine* over $\mathcal{L}$ is a triple $\mathcal{M} = (\mathcal{T}, \mathcal{I}, \mathcal{O})$ where:

- $\mathcal{T}$ is a finite set of sentences of $\mathcal{L}$.
  (this provides an abstract description for the machine)
- $\mathcal{I}$ is a set of distinct atomic sentences of $\mathcal{L}$.
  (this is the set of possible inputs to the machine)
- $\mathcal{O}$ is a set of distinct atomic sentences of $\mathcal{L}$.
  (this is the set of possible outputs from the machine)

(Atomic sentences are quantifier free sentences of the form $\bigwedge_{i=1}^{N} \pm R_i(\tau_{i1}, \ldots, \tau_{ir_i})$ for relations R_i and terms τ_{ik}.)

Theory Machines

Definition: Let $\mathcal{L}$ be a finite first-order language, a *theory machine* over $\mathcal{L}$ is a triple $\mathcal{M} = (\mathcal{T}, \mathcal{I}, \mathcal{O})$ where:

- $\mathcal{T}$ is a finite set of sentences of $\mathcal{L}$.
  (this provides an abstract description for the machine)
- $\mathcal{I}$ is a set of distinct atomic sentences of $\mathcal{L}$.
  (this is the set of possible inputs to the machine)
- $\mathcal{O}$ is a set of distinct atomic sentences of $\mathcal{L}$.
  (this is the set of possible outputs from the machine)

(Atomic sentences are quantifier free sentences of the form $\bigwedge_{i=1}^{N} \pm R_i(\tau_{i1}, \ldots, \tau_{ir_i})$ for relations R_i and terms τ_{ik}.)

Theory Machines

Definition: Let $\mathcal{L}$ be a finite first-order language, a *theory machine* over $\mathcal{L}$ is a triple $\mathcal{M} = (\mathcal{T}, \mathcal{I}, \mathcal{O})$ where:

- $\mathcal{T}$ is a finite set of sentences of $\mathcal{L}$.
  (this provides an abstract description for the machine)
- $\mathcal{I}$ is a set of distinct atomic sentences of $\mathcal{L}$.
  (this is the set of possible inputs to the machine)
- $\mathcal{O}$ is a set of distinct atomic sentences of $\mathcal{L}$.
  (this is the set of possible outputs from the machine)

(Atomic sentences are quantifier free sentences of the form $\bigwedge_{i=1}^{N} \pm R_i(\tau_{i1}, \ldots, \tau_{ir_i})$ for relations R_i and terms τ_{ik}.)

Examples of theory machines

We can describe a Turing machine M via a theory machine $\mathcal{M}_M = (\mathcal{T}_M, \mathcal{I}_M, \mathcal{O}_M)$ where:

- $\mathcal{T}_M = PA + PA_{\mathbb{Z}} + \text{Rules of } M$,
- $\mathcal{I}_M =$ All possible input words encoded as sentences,
- $\mathcal{O}_M = \{R(h), \neg R(h)\}$.

By modifying PA and $PA_{\mathbb{Z}}$ in $\mathcal{T}_M$ so that they include end-points, we can describe a theory machine that has finite models whose domain size grows as the input length increases.

Examples of theory machines

We can describe a Turing machine M via a theory machine $\mathcal{M}_M = (\mathcal{T}_M, \mathcal{I}_M, \mathcal{O}_M)$ where:

- $\mathcal{T}_M = PA + PA_{\mathbb{Z}} + \text{Rules of } M$,
- $\mathcal{I}_M =$ All possible input words encoded as sentences,
- $\mathcal{O}_M = \{R(h), \neg R(h)\}$.

By modifying PA and $PA_{\mathbb{Z}}$ in $\mathcal{T}_M$ so that they include end-points, we can describe a theory machine that has finite models whose domain size grows as the input length increases.

Examples of theory machines

We can describe a Turing machine M via a theory machine $\mathcal{M}_M = (\mathcal{T}_M, \mathcal{I}_M, \mathcal{O}_M)$ where:

- $\mathcal{T}_M = PA + PA_{\mathbb{Z}} + \text{Rules of } M$,
- $\mathcal{I}_M = \text{All possible input words encoded as sentences}$,
- $\mathcal{O}_M = \{R(h), \neg R(h)\}$.

By modifying PA and $PA_{\mathbb{Z}}$ in $\mathcal{T}_M$ so that they include end-points, we can describe a theory machine that has finite models whose domain size grows as the input length increases.

Examples of theory machines

We can describe a device N acting smoothly on $S \subseteq \mathbb{R}^n$ with time axis $[0, \infty)$ via a theory machine $\mathcal{M}_N = (\mathcal{T}_N, \mathcal{I}_N, \mathcal{O}_N)$ where:

- $\mathcal{T}_N = \mathbb{R}^n$ axioms $+ [0, \infty)$ axioms $+$ Evolution theory of N,
- $\mathcal{I}_N$ = Finite descriptions of relations on S at time 0,
- $\mathcal{O}_N$ = Finite descriptions of relations on S at some halting time.

If for each given input S is finite, then we can simulate N by finite models by modifying $\mathcal{T}_N$ so that it is satisfied by rational approximations to $\mathbb{R}^n \times [0, \infty)$.

Examples of theory machines

We can describe a device N acting smoothly on $S \subseteq \mathbb{R}^n$ with time axis $[0, \infty)$ via a theory machine $\mathcal{M}_N = (\mathcal{T}_N, \mathcal{I}_N, \mathcal{O}_N)$ where:

- $\mathcal{T}_N = \mathbb{R}^n$ axioms $+ [0, \infty)$ axioms $+$ Evolution theory of N,
- $\mathcal{I}_N$ = Finite descriptions of relations on S at time 0,
- $\mathcal{O}_N$ = Finite descriptions of relations on S at some halting time.

If for each given input S is finite, then we can simulate N by finite models by modifying $\mathcal{T}_N$ so that it is satisfied by rational approximations to $\mathbb{R}^n \times [0, \infty)$.

Examples of theory machines

We can describe a device N acting smoothly on $S \subseteq \mathbb{R}^n$ with time axis $[0, \infty)$ via a theory machine $\mathcal{M}_N = (\mathcal{T}_N, \mathcal{I}_N, \mathcal{O}_N)$ where:

- $\mathcal{T}_N = \mathbb{R}^n$ axioms $+ [0, \infty)$ axioms $+$ Evolution theory of N,
- $\mathcal{I}_N$ = Finite descriptions of relations on S at time 0,
- $\mathcal{O}_N$ = Finite descriptions of relations on S at some halting time.

If for each given input S is finite, then we can simulate N by finite models by modifying $\mathcal{T}_N$ so that it is satisfied by rational approximations to $\mathbb{R}^n \times [0, \infty)$.

Derivation

Definition: Let $\mathcal{M} = (\mathcal{T}, \mathcal{I}, \mathcal{O})$ be a theory machine, we say that $\mathcal{M}$ *derives* $\theta \in \mathcal{O}$ from $\phi \in \mathcal{I}$ if:

$$\mathcal{T} \cup \{\phi\} \text{ is satisfiable and } \mathcal{T} \cup \{\phi\} \models \theta.$$

(That is, in all models of $\mathcal{T}$ in which ϕ is true, θ must also be true, provided that such models exist).
We denote the set of such inputs by:

$$\mathcal{M}_\theta = \{\phi \in \mathcal{I} \mid \mathcal{T} \cup \{\phi\} \text{ is satisfiable and } \mathcal{T} \cup \{\phi\} \models \theta\},$$

which we call the *derivation set* of θ in $\mathcal{M}$.

Derivation

Definition: Let $\mathcal{M} = (\mathcal{T}, \mathcal{I}, \mathcal{O})$ be a theory machine, we say that $\mathcal{M}$ *derives* $\theta \in \mathcal{O}$ from $\phi \in \mathcal{I}$ if:

$$\mathcal{T} \cup \{\phi\} \text{ is satisfiable and } \mathcal{T} \cup \{\phi\} \models \theta.$$

(That is, in all models of $\mathcal{T}$ in which ϕ is true, θ must also be true, provided that such models exist).
We denote the set of such inputs by:

$$\mathcal{M}_\theta = \{\phi \in \mathcal{I} \mid \mathcal{T} \cup \{\phi\} \text{ is satisfiable and } \mathcal{T} \cup \{\phi\} \models \theta\},$$

which we call the *derivation set* of θ in $\mathcal{M}$.

Derivation

Definition: Let $\mathcal{M} = (\mathcal{T}, \mathcal{I}, \mathcal{O})$ be a theory machine, we say that $\mathcal{M}$ *derives* $\theta \in \mathcal{O}$ from $\phi \in \mathcal{I}$ if:

$$\mathcal{T} \cup \{\phi\} \text{ is satisfiable and } \mathcal{T} \cup \{\phi\} \models \theta.$$

(That is, in all models of $\mathcal{T}$ in which ϕ is true, θ must also be true, provided that such models exist).
We denote the set of such inputs by:

$$\mathcal{M}_\theta = \{\phi \in \mathcal{I} \mid \mathcal{T} \cup \{\phi\} \text{ is satisfiable and } \mathcal{T} \cup \{\phi\} \models \theta\},$$

which we call the *derivation set* of θ in $\mathcal{M}$.

Derivation of Word Problems

We can represent any word $w \in \{0,1\}^*$ where $w = a_1 \dots a_m$, as an atomic sentence of the form:

$$\rho_w = \bigwedge_{i=0}^{m} R^{a_i}(\triangleright^i(c)) \wedge W(\triangleright^m(c)) \wedge \neg W(\triangleright^{m+1}(c)).$$

Where R and W are a unary relations, $\triangleright$ is a unary function and c is a constant symbol.

We call ρ_w the *sentence representation of w* and denote the set of such representations by:

$$\mathcal{SR}_{\{0,1\}^*} = \{\rho_w \mid w \in \{0,1\}^*\}.$$

Derivation of Word Problems

We can represent any word $w \in \{0,1\}^*$ where $w = a_1 \ldots a_m$, as an atomic sentence of the form:

$$\rho_w = \bigwedge_{i=0}^{m} R^{a_i}(\triangleright^i(c)) \wedge W(\triangleright^m(c)) \wedge \neg W(\triangleright^{m+1}(c)).$$

Where R and W are a unary relations, $\triangleright$ is a unary function and c is a constant symbol.

We call ρ_w the *sentence representation of* w and denote the set of such representations by:

$$\mathcal{SR}_{\{0,1\}^*} = \{\rho_w \mid w \in \{0,1\}^*\}.$$

Return to the Example

We can describe a Turing machine M via a theory machine $\mathcal{M}_M = (\mathcal{T}_M, \mathcal{I}_M, \mathcal{O}_M)$ where:

- $\mathcal{T}_M = PA + PA_{\mathbb{Z}} + \text{Rules of } M$,
- $\mathcal{I}_M =$ All possible input words encoded as sentences,
- $\mathcal{O}_M = \{R(h), \neg R(h)\}$.

Return to the Example

We can describe a Turing machine M via a theory machine $\mathcal{M}_M = (\mathcal{T}_M, \mathcal{I}_M, \mathcal{O}_M)$ where:

- $\mathcal{T}_M = PA + PA_{\mathbb{Z}} + \text{Rules of } M$,
- $\mathcal{I}_M = \mathcal{SR}_{\{0,1\}*}$,
- $\mathcal{O}_M = \{R(h), \neg R(h)\}$.

Derivation of Word Problems

Definition: Let $\mathcal{M} = (\mathcal{T}, \mathcal{SR}_{\{0,1\}^*}, \mathcal{O})$ be a theory machine with $\{\theta, \psi\} \subseteq \mathcal{O}$. A decision problem $A \subseteq \{0,1\}^*$ is *totally derived* by $\mathcal{M}$ if:

$$\mathcal{M}_\theta = \{\rho_w \mid w \in A\} \text{ and } \mathcal{M}_\psi = \{\rho_w \mid w \in \{0,1\}^* \setminus A\}.$$

(So given any model $\mathfrak{A}$ of $\mathcal{T} \cup \{\rho_w\}$, we can check $\mathfrak{A} \models \theta$ to know $w \in A$ and check $\mathfrak{A} \models \psi$ to know $w \notin A$).
A decision problem $B \subseteq \{0,1\}^*$ is *partially derived* by $\mathcal{M}$ if:

$$\mathcal{M}_\theta = \{\rho_w \mid w \in B\}.$$

(So given any model $\mathfrak{B}$ of $\mathcal{T} \cup \{\rho_w\}$, if $\mathfrak{B} \models \theta$ then we know $w \in B$, but we do not necessarily have a sentence whose truth we can check to know that $w \notin B$).

Derivation of Word Problems

Definition: Let $\mathcal{M} = (\mathcal{T}, \mathcal{SR}_{\{0,1\}^*}, \mathcal{O})$ be a theory machine with $\{\theta, \psi\} \subseteq \mathcal{O}$. A decision problem $A \subseteq \{0,1\}^*$ is *totally derived* by $\mathcal{M}$ if:

$$\mathcal{M}_\theta = \{\rho_w \mid w \in A\} \text{ and } \mathcal{M}_\psi = \{\rho_w \mid w \in \{0,1\}^* \setminus A\}.$$

(So given any model $\mathfrak{A}$ of $\mathcal{T} \cup \{\rho_w\}$, we can check $\mathfrak{A} \models \theta$ to know $w \in A$ and check $\mathfrak{A} \models \psi$ to know $w \notin A$).
A decision problem $B \subseteq \{0,1\}^*$ is *partially derived* by $\mathcal{M}$ if:

$$\mathcal{M}_\theta = \{\rho_w \mid w \in B\}.$$

(So given any model $\mathfrak{B}$ of $\mathcal{T} \cup \{\rho_w\}$, if $\mathfrak{B} \models \theta$ then we know $w \in B$, but we do not necessarily have a sentence whose truth we can check to know that $w \notin B$).

Derivation of Word Problems

Definition: Let $\mathcal{M} = (\mathcal{T}, \mathcal{SR}_{\{0,1\}^*}, \mathcal{O})$ be a theory machine with $\{\theta, \psi\} \subseteq \mathcal{O}$. A decision problem $A \subseteq \{0,1\}^*$ is *totally derived* by $\mathcal{M}$ if:

$$\mathcal{M}_\theta = \{\rho_w \mid w \in A\} \text{ and } \mathcal{M}_\psi = \{\rho_w \mid w \in \{0,1\}^* \setminus A\}.$$

(So given any model $\mathfrak{A}$ of $\mathcal{T} \cup \{\rho_w\}$, we can check $\mathfrak{A} \models \theta$ to know $w \in A$ and check $\mathfrak{A} \models \psi$ to know $w \notin A$).

A decision problem $B \subseteq \{0,1\}^*$ is *partially derived* by $\mathcal{M}$ if:

$$\mathcal{M}_\theta = \{\rho_w \mid w \in B\}.$$

(So given any model $\mathfrak{B}$ of $\mathcal{T} \cup \{\rho_w\}$, if $\mathfrak{B} \models \theta$ then we know $w \in B$, but we do not necessarily have a sentence whose truth we can check to know that $w \notin B$).

Theory Machines vs Turing Machines

Theorem

A decision problem is computable by a Turing machine if and only if it is totally derivable.

Theorem

A decision problem can be computably enumerated by a Turing machine if and only if it is partially derivable.

(This analogously holds for computation with type-2 machine Turing machines an infinite-input theory machines.)

Theory Machines vs Turing Machines

Theorem

A decision problem is computable by a Turing machine if and only if it is totally derivable.

Theorem

A decision problem can be computably enumerated by a Turing machine if and only if it is partially derivable.

(This analogously holds for computation with type-2 machine Turing machines an infinite-input theory machines.)

Theory Machines vs Turing Machines

Theorem

A decision problem is computable by a Turing machine if and only if it is totally derivable.

Theorem

A decision problem can be computably enumerated by a Turing machine if and only if it is partially derivable.

(This analogously holds for computation with type-2 machine Turing machines an infinite-input theory machines.)

Theory Machines vs Turing Machines

Theorem

A decision problem is computable by a Turing machine if and only if it is totally derivable.

Theorem

A decision problem can be computably enumerated by a Turing machine if and only if it is partially derivable.

(This analogously holds for computation with type-2 machine Turing machines an infinite-input theory machines.)

Theory Machine Complexity

Observations on Resource Usage

Observe how we can represent an entire Turing machine computation that takes t time steps and utilises s tape squares, by a $t \times s$ sized rectangle. This computation can thus be described by a logical structure $\mathfrak{A}$ with domain size $||\mathfrak{A}|| = O(ts)$.

Similarly, to simulate a computation on $S \subset \mathbb{R}^n$ with precision ϵ and in time τ requires a structure $\mathfrak{B}$ with domain size $||\mathfrak{B}|| = O\left(\frac{|S|\tau}{\epsilon^{n+1}}\right)$.

Observations on Resource Usage

Observe how we can represent an entire Turing machine computation that takes t time steps and utilises s tape squares, by a $t \times s$ sized rectangle. This computation can thus be described by a logical structure $\mathfrak{A}$ with domain size $||\mathfrak{A}|| = O(ts)$.

Similarly, to simulate a computation on $S \subset \mathbb{R}^n$ with precision ϵ and in time τ requires a structure $\mathfrak{B}$ with domain size $||\mathfrak{B}|| = O\left(\frac{|S|\tau}{\epsilon^{n+1}}\right)$.

Derivation with Finite Resources

Definition: Let $\mathcal{M} = (\mathcal{T}, \mathcal{I}, \mathcal{O})$ be a theory machine, $\theta \in \mathcal{O}$ and $f : \mathbb{N} \to \mathbb{N}$ be an increasing function.

We say that $\mathcal{M}$ *derives* θ *with* f *resources* from $\phi \in \mathcal{I}$ if $\mathcal{T} \cup \{\phi\} \models \theta$ and there exists a model $\mathfrak{A}$ of $\mathcal{T} \cup \{\phi\}$ such that $||\mathfrak{A}|| \leqslant f(|\phi|)$.

(So from ϕ we can derive θ with a model of size of order $f(n)$ where n is the number of symbols in ϕ).

We denote the set of such elements of $\mathcal{I}$ by:

$$\mathcal{M}^f_\theta = \{\phi \in \mathcal{M}_\theta \mid \exists \mathfrak{A} : \mathfrak{A} \models \mathcal{T} \cup \{\phi\} \text{ and } ||\mathfrak{A}|| \leqslant f(|\phi|)\},$$

which we call the *f derivation set* of θ in $\mathcal{M}$.

Derivation with Finite Resources

Definition: Let $\mathcal{M} = (\mathcal{T}, \mathcal{I}, \mathcal{O})$ be a theory machine, $\theta \in \mathcal{O}$ and $f : \mathbb{N} \to \mathbb{N}$ be an increasing function.
We say that $\mathcal{M}$ *derives* θ *with* f *resources* from $\phi \in \mathcal{I}$ if $\mathcal{T} \cup \{\phi\} \models \theta$ and there exists a model $\mathfrak{A}$ of $\mathcal{T} \cup \{\phi\}$ such that $||\mathfrak{A}|| \leqslant f(|\phi|)$.
(So from ϕ we can derive θ with a model of size of order $f(n)$ where n is the number of symbols in ϕ).
We denote the set of such elements of $\mathcal{I}$ by:

$$\mathcal{M}_\theta^f = \{\phi \in \mathcal{M}_\theta \mid \exists \mathfrak{A} : \mathfrak{A} \models \mathcal{T} \cup \{\phi\} \text{ and } ||\mathfrak{A}|| \leqslant f(|\phi|)\},$$

which we call the f *derivation set* of θ in $\mathcal{M}$.

Derivation with Finite Resources

Definition: Let $\mathcal{M} = (\mathcal{T}, \mathcal{I}, \mathcal{O})$ be a theory machine, $\theta \in \mathcal{O}$ and $f : \mathbb{N} \to \mathbb{N}$ be an increasing function.
We say that $\mathcal{M}$ *derives* θ *with* f *resources* from $\phi \in \mathcal{I}$ if $\mathcal{T} \cup \{\phi\} \models \theta$ and there exists a model $\mathfrak{A}$ of $\mathcal{T} \cup \{\phi\}$ such that $||\mathfrak{A}|| \leqslant f(|\phi|)$.
(So from ϕ we can derive θ with a model of size of order $f(n)$ where n is the number of symbols in ϕ).
We denote the set of such elements of $\mathcal{I}$ by:

$$\mathcal{M}_\theta^f = \{\phi \in \mathcal{M}_\theta \mid \exists \mathfrak{A} : \mathfrak{A} \models \mathcal{T} \cup \{\phi\} \text{ and } ||\mathfrak{A}|| \leqslant f(|\phi|)\},$$

which we call the f *derivation set* of θ in $\mathcal{M}$.

Observations on Input Size

Observe how we can represent any word $w \in \{0,1\}^*$ by an atomic sentence $\rho_w = \bigwedge_{i=0}^{m} R^{a_i}(\triangleright^i(c)) \wedge W(\triangleright^m(c)) \wedge \neg W(\triangleright^{m+1}(c))$, which is of length $O(|w|^2)$.

Similarly note how we are able to represent a rational number r with a binary expansion of $r = b_{\text{-}l} \ldots b_{\text{-}1} b_0 . b_1 \ldots b_m$ with an atomic sentence of the form:

$$R(f_{\div 2}(f_{+1}^{b_{\text{-}l}}(\cdots f_{\div 2}(f_{+1}^{b_m}(f_{\times 2}^{l+m}(1)))\cdots))),$$

which is of length $O(|r|)$.

Observations on Input Size

Observe how we can represent any word $w \in \{0,1\}^*$ by an atomic sentence $\rho_w = \bigwedge_{i=0}^{m} R^{a_i}(\triangleright^i(c)) \wedge W(\triangleright^m(c)) \wedge \neg W(\triangleright^{m+1}(c))$, which is of length $O(|w|^2)$.

Similarly note how we are able to represent a rational number r with a binary expansion of $r = b_{\text{-}l} \ldots b_{\text{-}1} b_0 . b_1 \ldots b_m$ with an atomic sentence of the form:

$$R(f_{\div 2}(f_{+1}^{b_{\text{-}l}}(\cdots f_{\div 2}(f_{+1}^{b_m}(f_{\times 2}^{l+m}(1)))) \cdots))),$$

which is of length $O(|r|)$.

Efficient Derivation

Definition: Let $\mathcal{M} = (\mathcal{T}, \mathcal{SR}_{\{0,1\}^*}, \mathcal{O})$ be a theory machine with $\{\theta, \psi\} \subseteq \mathcal{O}$. A decision problem $A \subseteq \{0,1\}^*$ is *totally derived by $\mathcal{M}$ with polynomial resources* if there is a polynomial function $p : \mathbb{N} \to \mathbb{N}$ such that:

$$\mathcal{M}^p_\theta = \{\rho_w \mid w \in A\}, \text{ and } \mathcal{M}^p_\psi = \{\rho_w \mid w \in \{0,1\}^* \setminus A\}.$$

(So we can always find a polynomial sized model $\mathfrak{A}$ of $\mathcal{T} \cup \{\rho_w\}$, and then check whether $\mathfrak{A} \models \theta$ or $\mathfrak{A} \models \psi$ to know that $w \in A$ or $w \notin A$). A decision problem $B \subseteq \{0,1\}^*$ is *partially derived by $\mathcal{M}$ with polynomial resources* if:

$$\mathcal{M}^p_\theta = \{\rho_w \mid w \in B\}.$$

(So if $w \in B$ then we can find a polynomial sized model of $\mathcal{T} \cup \{\rho_w\}$ in which θ is true, but otherwise such a model may not exist).

Efficient Derivation

Definition: Let $\mathcal{M} = (\mathcal{T}, \mathcal{SR}_{\{0,1\}^*}, \mathcal{O})$ be a theory machine with $\{\theta, \psi\} \subseteq \mathcal{O}$. A decision problem $A \subseteq \{0,1\}^*$ is *totally derived by $\mathcal{M}$ with polynomial resources* if there is a polynomial function $p : \mathbb{N} \to \mathbb{N}$ such that:

$$\mathcal{M}^p_\theta = \{\rho_w \mid w \in A\}, \text{ and } \mathcal{M}^p_\psi = \{\rho_w \mid w \in \{0,1\}^* \setminus A\}.$$

(So we can always find a polynomial sized model $\mathfrak{A}$ of $\mathcal{T} \cup \{\rho_w\}$, and then check whether $\mathfrak{A} \models \theta$ or $\mathfrak{A} \models \psi$ to know that $w \in A$ or $w \notin A$).

A decision problem $B \subseteq \{0,1\}^*$ is *partially derived by $\mathcal{M}$ with polynomial resources* if:

$$\mathcal{M}^p_\theta = \{\rho_w \mid w \in B\}.$$

(So if $w \in B$ then we can find a polynomial sized model of $\mathcal{T} \cup \{\rho_w\}$ in which θ is true, but otherwise such a model may not exist).

Efficient Derivation

Definition: Let $\mathcal{M} = (\mathcal{T}, \mathcal{SR}_{\{0,1\}^*}, \mathcal{O})$ be a theory machine with $\{\theta, \psi\} \subseteq \mathcal{O}$. A decision problem $A \subseteq \{0,1\}^*$ is *totally derived by $\mathcal{M}$ with polynomial resources* if there is a polynomial function $p : \mathbb{N} \to \mathbb{N}$ such that:

$$\mathcal{M}^p_\theta = \{\rho_w \mid w \in A\}, \text{ and } \mathcal{M}^p_\psi = \{\rho_w \mid w \in \{0,1\}^* \setminus A\}.$$

(So we can always find a polynomial sized model $\mathfrak{A}$ of $\mathcal{T} \cup \{\rho_w\}$, and then check whether $\mathfrak{A} \models \theta$ or $\mathfrak{A} \models \psi$ to know that $w \in A$ or $w \notin A$).

A decision problem $B \subseteq \{0,1\}^*$ is *partially derived by $\mathcal{M}$ with polynomial resources* if:

$$\mathcal{M}^p_\theta = \{\rho_w \mid w \in B\}.$$

(So if $w \in B$ then we can find a polynomial sized model of $\mathcal{T} \cup \{\rho_w\}$ in which θ is true, but otherwise such a model may not exist).

Examples

- As any problem in P can be decided by a Turing machine in polynomial time and space such problems are totally derivable by a theory machine with polynomial resources.
- Similarly, since any accepting computation path of a non-deterministic Turing machine utilises polynomial time and space, any problem in NP must be partially derivable by a theory machine with polynomial resources.
- If a Newtonian kinematic system can decide a problem whose space, time and precision requirements grow polynomially with the input length, then such a problem can be totally derived by a theory machine with polynomial resources.

Examples

- As any problem in P can be decided by a Turing machine in polynomial time and space such problems are totally derivable by a theory machine with polynomial resources.
- Similarly, since any accepting computation path of a non-deterministic Turing machine utilises polynomial time and space, any problem in NP must be partially derivable by a theory machine with polynomial resources.
- If a Newtonian kinematic system can decide a problem whose space, time and precision requirements grow polynomially with the input length, then such a problem can be totally derived by a theory machine with polynomial resources.

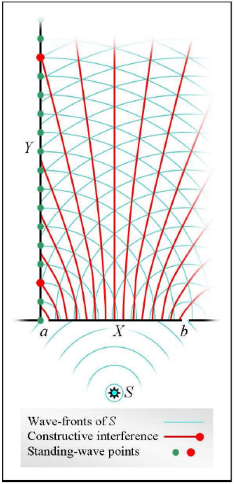
Examples

- As any problem in P can be decided by a Turing machine in polynomial time and space such problems are totally derivable by a theory machine with polynomial resources.
- Similarly, since any accepting computation path of a non-deterministic Turing machine utilises polynomial time and space, any problem in NP must be partially derivable by a theory machine with polynomial resources.
- If a Newtonian kinematic system can decide a problem whose space, time and precision requirements grow polynomially with the input length, then such a problem can be totally derived by a theory machine with polynomial resources.

Non-example

Ed Blakey's double slit factoriser (→) (which uses classical physics) is polynomially bounded in space and time, but the operational precision it requires grows exponentially with the size of the input.

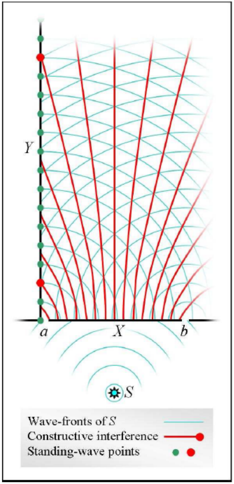
Hence the machine's models are required to grow exponentially in size with the length of the input and the problem can only be totally derived here with exponential resources.



Source: Ed Blakey

Non-example

Ed Blakey's double slit factoriser (→) (which uses classical physics) is polynomially bounded in space and time, but the operational precision it requires grows exponentially with the size of the input.
Hence the machine's models are required to grow exponentially in size with the length of the input and the problem can only be totally derived here with exponential resources.



Source: Ed Blakey

The Limits of Feasible Derivation

Theorem

A decision problem is totally derivable by a theory machine with polynomial resources if and only if it is in $NP \cap$ co-NP.

Theorem

A decision problem is partially derivable by a theory machine with polynomial resources if and only if it is in NP.

The Limits of Feasible Derivation

Theorem

A decision problem is totally derivable by a theory machine with polynomial resources if and only if it is in $NP \cap$ co-NP.

Theorem

A decision problem is partially derivable by a theory machine with polynomial resources if and only if it is in NP.

The Limits of Feasible Derivation

Theorem

A decision problem is totally derivable by a theory machine with polynomial resources if and only if it is in $NP \cap$ co-NP.

Theorem

A decision problem is partially derivable by a theory machine with polynomial resources if and only if it is in NP.

The Limits of Feasible Derivation

Proof (outline): $(\Rightarrow)$ Follows from the fact that in NP we can non-deterministically generate a polynomially-sized model $\mathfrak{A}$ of $\mathcal{T} \cup \{\rho_w\}$ and such a model is guaranteed to exist.

Given $\mathfrak{A}$ we can then efficiently check whether $\mathfrak{A} \models \theta$ or $\mathfrak{A} \models \psi$.

The Limits of Feasible Derivation

Proof (outline): ($\Leftarrow$) If $A \in NP \cap \text{co-}NP$ then there exists two Turing machines M_1 and M_2 that non-deterministic compute in polynomial time A and $\{0,1\}^* \setminus A$ respectively.

We can then construct a theory machine which can carry out a computation of either M_1 or M_2, but, by virtue of its theory, is prevented from reaching a halt and reject configuration. Thus it only ever produces an accepting computation on the appropriate machine. $\square$

The Limits of Feasible Derivation

Proof (outline): ($\Leftarrow$) If $A \in NP \cap \text{co-}NP$ then there exists two Turing machines M_1 and M_2 that non-deterministic compute in polynomial time A and $\{0,1\}^* \setminus A$ respectively.

We can then construct a theory machine which can carry out a computation of either M_1 or M_2, but, by virtue of its theory, is prevented from reaching a halt and reject configuration. Thus it only ever produces an accepting computation on the appropriate machine. $\square$

Conclusion

On Computation and Logical Consequence Again

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
Total derivation with a Theory machine	$\Longleftrightarrow$	The Turing machine model of computation
$\downarrow$		$\downarrow$
Total derivation with polynomial resources	$\not\Longleftrightarrow$ (if $P \neq NP \cap co\text{-}NP$)	Sequential algorithmic basis for complexity

On Computation and Logical Consequence Again

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
Total derivation with a Theory machine	$\Longleftrightarrow$	The Turing machine model of computation
$\downarrow$		$\downarrow$
Total derivation with polynomial resources	$\not\Longleftrightarrow$ (if $P \neq NP \cap co\text{-}NP$)	Sequential algorithmic basis for complexity

On Computation and Logical Consequence Again

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
Total derivation with a Theory machine	$\Longleftrightarrow$	The Turing machine model of computation
$\downarrow$		$\downarrow$
Total derivation with polynomial resources	$\not\Longleftrightarrow$ (if $P \neq NP \cap co\text{-}NP$)	Sequential algorithmic basis for complexity

On Computation and Logical Consequence Again

$\mathcal{T} \models \theta$		$\mathcal{T} \vdash \theta$
Truth of θ in any model of $\mathcal{T}$	$\Longleftrightarrow$	Seqential logical proof of θ from $\mathcal{T}$
$\downarrow$		$\downarrow$
Total derivation with a Theory machine	$\Longleftrightarrow$	The Turing machine model of computation
$\downarrow$		$\downarrow$
Total derivation with polynomial resources	$\not\Longleftrightarrow$ (if $P \neq NP \cap co\text{-}NP$)	Sequential algorithmic basis for complexity

On Computation and Logical Consequence Again

$$
\begin{array}{ccc}
\mathcal{T} \models \theta & & \mathcal{T} \vdash \theta \\[1em]
\text{Truth of } \theta \text{ in} & & \text{Seqential logical} \\
\text{any model of } \mathcal{T} & \Longleftrightarrow & \text{proof of } \theta \text{ from } \mathcal{T} \\
\downarrow & & \downarrow \\
\text{Total derivation with} & & \text{The Turing machine} \\
\text{a Theory machine} & \Longleftrightarrow & \text{model of computation} \\
\downarrow & & \downarrow \\
\text{Total derivation with} & & \text{Sequential algorithmic} \\
\text{polynomial resources} & \not\Longleftrightarrow & \text{basis for complexity} \\
 & \text{(if } P \neq NP \cap co\text{-}NP\text{)} &
\end{array}
$$

What Might All This Mean?

The theory machine constructed to solve $NP \cap$ co-NP problems with polynomial resources did so by essentially violating causality. As possible futures were able to influence present decisions, a computation path was pursued by only if such a path was able to eventually lead to an accepting configuration.

Is it causality violation that enables quantum computers to efficiently decide problems in $(NP \cap \text{co-}NP) \setminus P$?

If so, does this mean that there exists an alternative description of quantum theory whose models grow polynomially with time and the number of qubits?

What Might All This Mean?

The theory machine constructed to solve $NP \cap \text{co-}NP$ problems with polynomial resources did so by essentially violating causality. As possible futures were able to influence present decisions, a computation path was pursued by only if such a path was able to eventually lead to an accepting configuration.

Is it causality violation that enables quantum computers to efficiently decide problems in $(NP \cap \text{co-}NP) \setminus P$?

If so, does this mean that there exists an alternative description of quantum theory whose models grow polynomially with time and the number of qubits?

What Might All This Mean?

The theory machine constructed to solve $NP \cap \text{co-}NP$ problems with polynomial resources did so by essentially violating causality. As possible futures were able to influence present decisions, a computation path was pursued by only if such a path was able to eventually lead to an accepting configuration.

Is it causality violation that enables quantum computers to efficiently decide problems in $(NP \cap \text{co-}NP) \setminus P$?

If so, does this mean that there exists an alternative description of quantum theory whose models grow polynomially with time and the number of qubits?

Further Work

- Generalise the concept of theory machine derivation to describe probabilistic derivation.
- Look into describing quantum computation with probabilistic theory machines.
- Look into what happens when theory machines are given higher-order theories.
- Are there new sorts of exotic computation devices that are best described with theory machines?

Further Work

- Generalise the concept of theory machine derivation to describe probabilistic derivation.
- Look into describing quantum computation with probabilistic theory machines.
- Look into what happens when theory machines are given higher-order theories.
- Are there new sorts of exotic computation devices that are best described with theory machines?

Further Work

- Generalise the concept of theory machine derivation to describe probabilistic derivation.
- Look into describing quantum computation with probabilistic theory machines.
- Look into what happens when theory machines are given higher-order theories.
- Are there new sorts of exotic computation devices that are best described with theory machines?

Further Work

- Generalise the concept of theory machine derivation to describe probabilistic derivation.
- Look into describing quantum computation with probabilistic theory machines.
- Look into what happens when theory machines are given higher-order theories.
- Are there new sorts of exotic computation devices that are best described with theory machines?