Comparing the Computational Power of Models of Computation

(*Models of Computation* Advanced Course, July 7–11, 2025)

Clemens Grabmayer

Department of Computer Science, Gran Sasso Science Institute

Abstract. For an abstract formulation of models of computations with their constitutive features, we sketch definitions of the two relations 'the computational power of a model $\mathcal{M}_1$ {is subsumed by | is equivalent to} that of a model $\mathcal{M}_2$'. Based on these relations we define 'Turing-completeness' and 'Turing-equivalence' for models of computation.

1 Preliminaries

Let $f : A \rightharpoonup B$ be a partial function. For all $a \in A$, we indicate by $f(a)\!\downarrow$ that f is defined on a, and by $f(a)\!\uparrow$ that f is undefined on a. By the *domain of* f we mean the set $dom(f) := \{a \in A : f(a)\!\downarrow\} \subseteq A$, and by the *range of* f we mean the set $ran(f) := \{f(a) : a \in A, f(a)\!\downarrow\} \subseteq B$. The partial function f is called *total* if $dom(f) = A$, that is, if f is defined for all elements of A.

2 Models of Computations, viewed abstractly

Definition 1. A*(n abstractly viewed) model of computation (MoC)* is a class $\mathcal{M}$ of machines/systems/... in which every member $M \in \mathcal{M}$ has the following constitutive features:

- M has a countably infinite set $I_{\mathcal{M}}$ of *input objects*, and a denumerable set $O_{\mathcal{M}}$ of *output objects* (note that both $I_{\mathcal{M}}$ and $O_{\mathcal{M}}$ are specific to the model $\mathcal{M}$, and do not depend on the specific member $M \in \mathcal{M}$);
- M has a set C_M of *configurations*, which contains the subset $EC_M \subseteq C_M$ of *end-configurations* of M;
- M has an injective function $\alpha_M : I_M \to C_M$, which maps input objects of $\mathcal{M}$ to configurations of M, and is required to be intuitively computable easily;
- M defines a *one-step computation relation* $\mapsto_M$ on the set C_M; the transitive closure of $\mapsto_M$ is designated by $\mapsto_M^*$;
- M has a partial function $\omega_M : EC_M \rightharpoonup O_M$, which maps *some* end-configurations of M to output objects of $\mathcal{M}$; the partial function ω_M is 'easily' computable, and furthermore, membership of end-configurations in $dom(\omega_M)$ is required to be easily decidable intuitively.

We say that $\mathcal{M}$ is *deterministic*, if for all $M \in \mathcal{M}$ the one-step computation relation $\mapsto_M$ is deterministic: for all configurations $c \in C_M$ there is at most one configuration $c' \in C_M$ such that $c \mapsto_M c'$.

Functions, and partial functions, defined by a member of a deterministic MoC can be defined as follows.

Definition 2. Let $\mathcal{M}$ be a deterministic model of computation.

Let $M \in \mathcal{M}$, and let $f : I_{\mathcal{M}} \rightharpoonup O_{\mathcal{M}}$ be a partial function. We say that M *computes* f if the following two statements hold for all $x \in I_{\mathcal{M}}$:

$$f(x)\!\downarrow \implies \exists c \in EC_M \left[\alpha_M(x) \mapsto_M^* c \ \wedge \ \omega_M(c)\!\downarrow \ \wedge \ \omega_M(c) = f(x) \right],$$
$$f(x)\!\uparrow \implies \neg \exists c \in EC_M \left[\alpha_M(x) \mapsto_M^* c \ \wedge \ \omega_M(c)\!\downarrow \right].$$

The set of *functions* $\mathcal{F}(M)$ that are *computable by* M are then defined as:

$$\mathcal{F}(M) := \left\{ f : I_{\mathcal{M}} \rightharpoonup O_{\mathcal{M}} \;\middle|\; M \text{ computes } f \right\}$$

Finally we define the set $\mathcal{F}(\mathcal{M})$ of *computable functions of the MoC* $\mathcal{M}$ by:

$$\mathcal{F}(\mathcal{M}) := \bigcup_{M \in \mathcal{M}} \mathcal{F}(M) = \left\{ f : I_{\mathcal{M}} \rightharpoonup O_{\mathcal{M}} \;\middle|\; (\exists M \in \mathcal{M})\big[M \text{ computes } f\big] \right\}.$$

This definition also stipulates computability for total functions, namely as the computability of partial functions that happen to be total. However, for total functions the defining condition can obviously be simplified as stated by the proposition below.

Proposition 3. *Let $\mathcal{M}$ be a deterministic model of computation, and $M \in \mathcal{M}$. Let $f : I_{\mathcal{M}} \to O_{\mathcal{M}}$ be a function. Then M computes the function f if it holds:*

$$\forall x \in I_{\mathcal{M}} \, \exists c \in EC_M \left[\alpha_M(x) \mapsto_M^* c \ \wedge \ \omega_M(c)\!\downarrow \ \wedge \ \omega_M(c) = f(x) \right].$$

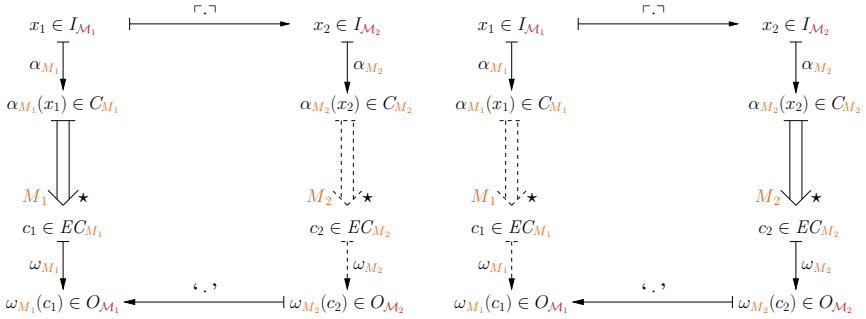
3 Reductions between MoC's

Let $I_{\mathcal{M}_1}$ and $I_{\mathcal{M}_2}$ be the set of input objects, and $O_{\mathcal{M}_1}$ and $O_{\mathcal{M}_2}$ be the set of output objects of two MoC's $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. By a *pair of coding and decoding functions between* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ we mean a pair $\langle \ulcorner \cdot \urcorner, \text{`} \cdot \text{'} \rangle$ that consists of a bijective function $\ulcorner \cdot \urcorner : I_{\mathcal{M}_1} \to I_{\mathcal{M}_2}$ from the input objects of $\mathcal{M}_1$ to input objects of $\mathcal{M}_2$, and a bijective function $\text{`} \cdot \text{'} : O_{\mathcal{M}_2} \to O_{\mathcal{M}_1}$ from the output objects of $\mathcal{M}_2$ to the output objects of $\mathcal{M}_1$.

Now we define what it means that two machines/systems/... 'simulate each other' with respect to given coding/encoding functions.

Definition 4. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be MoC's. Let $\langle \ulcorner \cdot \urcorner, \text{`} \cdot \text{'} \rangle$ be a pair of encoding and decoding functions between $\mathcal{M}_1$ and $\mathcal{M}_2$. Let $M_1 \in \mathcal{M}_1$ and $M_2 \in \mathcal{M}_2$ be machines/systems/... in these MoC's.

We say that M_1 *and* M_2 *simulate each other with respect to* $\langle \ulcorner \cdot \urcorner, \text{`} \cdot \text{'} \rangle$ if, for all $x_1 \in I_{\mathcal{M}_1}$, the $\forall\exists$-statements hold that correspond to the following two pictures:



(normal arrows signal assumed transitions, broken lines signal existence of transitions); more formally, the statement corresponding to the diagram on the left is:

$$\begin{aligned}&\forall x_1 \in I_{\mathcal{M}_1}\\ &\quad \forall c_1 \in EC_{M_1}\big[\,\alpha_{M_1}(x_1) \mapsto^*_{M_1} c_1 \ \wedge\ \omega_{M_1}(c_1)\!\downarrow \implies\\ &\qquad \exists c_2 \in EC_{M_2}\big(\,\alpha_{M_2}(\ulcorner x_1 \urcorner) \mapsto^*_{M_2} c_2 \ \wedge\ \omega_{M_2}(c_2)\!\downarrow \ \wedge\ \omega_{M_1}(c_1) = \text{`}\omega_{M_2}(c_2)\text{'}\,\big)\big]\,.\end{aligned}$$

The statement corresponding to the diagram on the right is the following:

$$\begin{aligned}&\forall x_1 \in I_{\mathcal{M}_1}\\ &\quad \forall c_2 \in EC_{M_2}\big[\,\alpha_{M_2}(\ulcorner x_1 \urcorner) \mapsto^*_{M_2} c_2 \ \wedge\ \omega_{M_2}(c_2)\!\downarrow \implies\\ &\qquad \exists c_1 \in EC_{M_1}\big(\,\alpha_{M_1}(x_1) \mapsto^*_{M_1} c_1 \ \wedge\ \omega_{M_1}(c_1)\!\downarrow \ \wedge\ \omega_{M_1}(c_1) = \text{`}\omega_{M_2}(c_2)\text{'}\,\big)\big]\,.\end{aligned}$$

Using the notion of 'back-and-forth simulation' between members of MoC's formalised in Definition 4, we can now define a preorder relation that reduces/subsumes the computational power of one MoC to/by that of another MoC.

Definition 5. For MoC's $\mathcal{M}_1$ and $\mathcal{M}_2$ we define the following concepts of relative subsumption, subsumption, and equivalence of computational power:

(i) Let $\langle \ulcorner\cdot\urcorner, \text{`}\cdot\text{'}\rangle$ be a pair of encoding and decoding functions between $\mathcal{M}_1$ and $\mathcal{M}_2$. We say that *the computational power of* $\mathcal{M}_1$ *is subsumed by that of* $\mathcal{M}_2$ *with respect to coding via* $\ulcorner\cdot\urcorner$ *and decoding via* $\text{`}\cdot\text{'}$, denoted symbolically by $\mathcal{M}_1 \leq_{\langle \ulcorner\cdot\urcorner,\, \text{`}\cdot\text{'}\rangle} \mathcal{M}_2$, if it holds:

$$\begin{aligned}&(\forall M_1 \in \mathcal{M}_1)\,(\exists M_2 \in \mathcal{M}_2)\\ &\qquad \big[\,M_1 \text{ and } M_2 \text{ simulate each other with respect to } \langle \ulcorner\cdot\urcorner, \text{`}\cdot\text{'}\rangle\,\big]\,.\end{aligned}$$

(ii) We say that *the computational power of* $\mathcal{M}_1$ *is subsumed by that of* $\mathcal{M}_2$, denoted symbolically by $\mathcal{M}_1 \leq \mathcal{M}_2$, if:

$$\begin{aligned}&(\exists \text{ a pair } \langle \ulcorner\cdot\urcorner, \text{`}\cdot\text{'}\rangle \text{ of informally computable encoding and decoding}\\ &\quad \text{functions from } I_{\mathcal{M}_1} \text{ to } I_{\mathcal{M}_2}, \text{ and from } O_{\mathcal{M}_2} \text{ to } O_{\mathcal{M}_1})\\ &\qquad \big[\,\mathcal{M}_1 \leq_{\langle \ulcorner\cdot\urcorner,\, \text{`}\cdot\text{'}\rangle} \mathcal{M}_2\,\big]\,.\end{aligned}$$

(iii) We say that *the computational power of* $\mathcal{M}_1$ *is equivalent to that of* $\mathcal{M}_2$, denoted by $\mathcal{M}_1 \sim \mathcal{M}_2$, if both $\mathcal{M}_1 \leq \mathcal{M}_2$ and $\mathcal{M}_2 \leq \mathcal{M}_1$ hold.

The following theorem guarantees that subsumption of computational power between MoC's preserves the computable functions up to coding and decoding. More precisely the theorem states that whenever the computational power of $\mathcal{M}_1$ is subsumed by that of $\mathcal{M}_2$ with respect to coding via $\ulcorner\cdot\urcorner$ and decoding via $\text{`}\cdot\text{'}$, then the functions that are computable by $\mathcal{M}_1$ are contained among the functions that are computable by $\mathcal{M}_2$ up to coding via $\ulcorner\cdot\urcorner$ and decoding via $\text{`}\cdot\text{'}$.

Theorem 6. *For all MoC's* $\mathcal{M}_1$ *and* $\mathcal{M}_2$*, and pairs* $\langle \ulcorner\cdot\urcorner, \text{`}\cdot\text{'} \rangle$ *of encoding and decoding functions between* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *the follwing statement holds:*

$$\mathcal{M}_1 \leq_{\langle \ulcorner\cdot\urcorner,\, \text{`}\cdot\text{'} \rangle} \mathcal{M}_2 \implies \mathcal{F}(\mathcal{M}_1) \subseteq \left\{ \text{`}\cdot\text{'} \circ f \circ \ulcorner\cdot\urcorner \;\middle|\; f \in \mathcal{F}(\mathcal{M}_2) \right\}.$$

4 Turing-completeness and Turing-equvalence of MoC's

Using the notion of 'computable reduction' between MoCs formalised in Definition 5, we can now define the notions of 'Turing-completeness' and 'Turing-equivalence' of a MoC.

As notation we will use the following. By $\mathcal{TM}(\Sigma)$ we mean the abstract model of computation that arises from Turing machines with input alphabet Σ by considering their features according to Definition 1 (and thus for example with $I_{\mathcal{M}} = O_{\mathcal{M}} = \Sigma^*$, the words over Σ).

Definition 7. Let $\mathcal{TM}$ be the class of Turing machines, and $\mathcal{M}$ an MoC.

We say that $\mathcal{M}$ is *Turing-complete* if $\mathcal{TM}(\Sigma) \leq \mathcal{M}$ for some alphabet Σ, that is, if the computational power of Turing machines over input alphabet alphabet Σ is subsumed by that of $\mathcal{M}$.

We say that $\mathcal{M}$ is *Turing-equivalent* if $\mathcal{M} \sim \mathcal{TM}(\Sigma)$ for some alphabet Σ, that is, if, for some alphabet Σ, both the computational power of Turing machines over input alphabet Σ is subsumed by that of $\mathcal{M}$, and the computational power of $\mathcal{M}$ is subsumed by that of Turing machines over input alphabet Σ.