Turing Categories and Incompleteness

Chad Nester

June 7, 2014

Incompleteness

First, the quick, abstract version of incompleteness

Incompleteness

given a logical system $\mathcal{L}$, let

- $g(E)$ be the Gödel number of the expression E
- E_n be such that $g(E_n) = n$
- $d(n) = g(E_n(n))$
- $\mathcal{P}$ be the set of provable sentences
- P be the set of Gödel numbers of provable sentences

Incompleteness

we say that $\mathcal{L}$ is *correct* if every provable sentence is true, and no refutable sentence is true.

Incompleteness

Now, define A^* by

$$n \in A^* \Leftrightarrow d(n) \in A$$

Incompleteness

Theorem

If $\mathcal{L}$ is correct and $(\overline{P})^*$ is expressible in $\mathcal{L}$ (where $\overline{P}$ is the complement of P), then there is a true sentence of $\mathcal{L}$ that is not provable in $\mathcal{L}$

Incompleteness

Proof

let H express $(\overline{P})^*$, and let $h = g(H)$. Since H expresses $(\overline{P})^*$,

$$
\begin{aligned}
H(h) \text{ is true } &\Leftrightarrow h \in \overline{P}^* \\
&\Leftrightarrow d(h) \in \overline{P} \\
&\Leftrightarrow d(h) \notin P \\
&\Leftrightarrow g(H(h)) \notin P \\
&\Leftrightarrow H(h) \notin \mathcal{P}
\end{aligned}
$$

so $H(h)$ is true if and only if $H(h)$ is not provable.

Incompleteness

Now, if $H(h)$ is false, then $H(h)$ is provable, but this is impossible as $\mathcal{L}$ is correct, which means all provable sentences are true! Since $H(h)$ cannot be false, it it true, and is thus not provable.

$\square$

Turing Categories

Next, we need to know what Turing categories are.

Turing Categories

A category $\mathbb{X}$ is a *restriction category* if for every map $f : A \to B$ in $\mathbb{X}$, there is a map $\overline{f} : A \to A$ in $\mathbb{X}$ satisfying

R1 $\overline{f} f = f$

R2 $\overline{f}\,\overline{g} = \overline{g}\,\overline{f}$

R3 $\overline{\overline{f} g} = \overline{f}\,\overline{g}$

R4 $f \overline{g} = \overline{fg} f$

we say $f : A \to B$ is *total* if $\overline{f} = 1_A$.

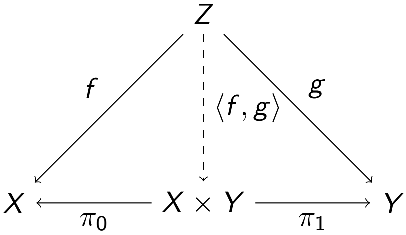
Turing Categories

Restriction categories come with a free partial order on every homset:

$$f \leq g \Leftrightarrow \overline{f} g = f$$

Turing Categories

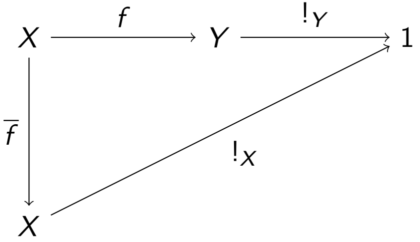
A restriction category $\mathbb{X}$ has *restriction products* if for every pair of objects X, Y in $\mathbb{X}$, there is an object $X \times Y$, the restriction product of X and Y, with maps π_0, π_1 such that for every $f : Z \to X$ and $g : Z \to Y$, there is a unique map $\langle f, g \rangle$ making the following diagram commute



and satisfying $\langle f, g \rangle \pi_0 = \overline{g} f$, $\langle f, g \rangle \pi_1 = \overline{f} g$.

Turing Categories

A restriction category $\mathbb{X}$ has a *restriction terminal object* if it contains an object 1 such that for every object A in $\mathbb{X}$ there is a unique total map $!_A : A \to 1$ such that for any $f : X \to Y$



Turing Categories

A restriction category with both restriction products and a restriction terminal object is called a *cartesian restriction category*.

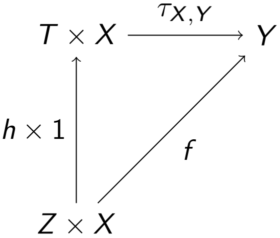
For Example: Partial functions on sets

- The restriction terminal object is $1 = \{*\}$, a one element set.
- The restriction product of A, B is the direct product, $A \times B$, with the usual projection.

Turing Categories

let $\mathbb{X}$ be a cartesian restriction category.

Given a map $\tau_{X,Y} : T \times X \to Y$, a morphism $f : Z \times X \to Y$ is said to admit a $\tau_{X,Y}$-*index* when there exists a total map $h : Z \to T$ making



commute. In this case, call h a $\tau_{X,Y}$-index for f.

$\tau_{X,Y}$ is called a *universal application* when every $f : Z \times X \to Y$ admits a $\tau_{X,Y}$-index.

A *Turing object* in an object T such that for each $X, Y \in \mathbb{X}$, there is a universal application $\tau_{X,Y} : T \times X \to Y$.

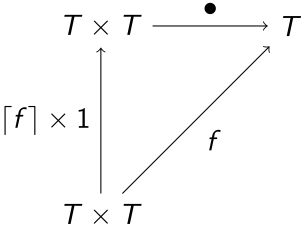
$\mathbb{X}$ is a *Turing category* if it posesses a Turing object.

equivalently. . .

Turing Categories

A *Turing category* is a cartesian restriction category with a distinguished object T, called the *Turing object*, such that

- every object is a retract of T
- there is an application map, $\bullet$, such that for every $f : T \times T \to T$ there is a total map $\lceil f \rceil : T \to T$ making the following diagram commute.

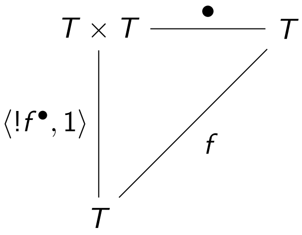


Turing Categories

Proposition

A cartesian restriction category $\mathbb{X}$ is a Turing category if it posesses an object T such that

- *every object is a retract of T*
- *there is a map $\bullet : T \times T \to T$ such that for all $f : T \to T$ there is a total point $f^\bullet : 1 \to T$ such that*



commutes.

Turing Categories

Turing Categories capture computability. For example

- λ-calculus gives a Turing category. (Take the reflexive object of the cartesian closed category as the Turing object)
- In fact, Turing catgories are in one-to-one correspondence with partial combinatory algebras

Incompleteness in Turing Categories

There is also a rather nice way to do Gödel incompleteness in a Turing category

Incompleteness in Turing Categories

Theorem

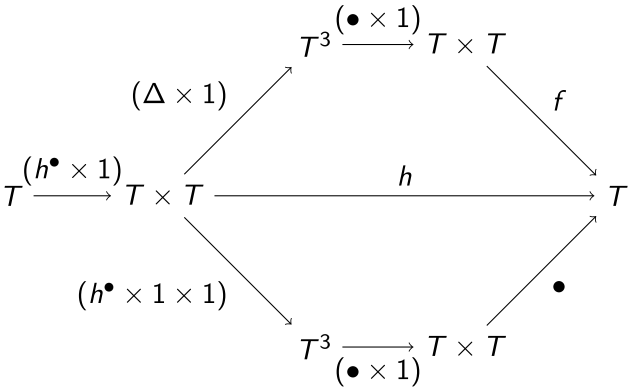
(second recursion theorem) In any Turing category, for any $f : T \times T \to T$ where T is the Turing object, there is a total point $p : 1 \to T$ such that $(p \times 1)\bullet = (p \times 1)f$.

Proof

let $h = (\Delta \times 1)(\bullet \times 1)f$. Then there is a code $h^\bullet$ for h with $(h^\bullet \times 1)\bullet$ total and $(h^\bullet \times 1 \times 1)(\bullet \times 1)\bullet = h$.

Setting $p = (h^\bullet \times h^\bullet)\bullet$ makes

Incompleteness in Turing Categories



commute, which gives $(p \times 1)\bullet = (p \times 1)f$, as required. □

Incompleteness in Turing Categories

In a Turing category with Turing structure $(T, \bullet)$, a *provability predicate* is a restriction idempotent $e_{pf} \in \mathcal{O}(T \times T)$ such that k_1, k_2 are in e_{pf} $(k_1 \vdash k_2)$ if and only if

$$\langle !k_1, 1 \rangle \bullet \leq \langle !k_2, 1 \rangle \bullet$$

Incompleteness in Turing Categories

Theorem

Every nontrivial Turing category with a restriction zero and a provability predicate has a predicate (subobject) of the terminal object which is neither $0 \in \mathcal{O}(1)$ *nor* $1 \in \mathcal{O}(1)$.

Incompleteness in Turing Categories

Proof

let $e_{pf} \in \mathcal{O}(T \times T)$ be the provability predicate, and let $0^\bullet$, $0^{\bullet\bullet}$ be codes for the maps 0 and $!0^\bullet$ respectively.

let $f = (\langle 1, !0^{\bullet\bullet} \rangle comp \times 1) e_{pf} !k$ for some $k : 1 \to T$.

by the second recursion theorem, there is a point $p^\bullet : 1 \to T$ such that $(p^\bullet \times 1)\bullet = (p^\bullet \times 1)f$. Note that $p^\bullet$ is a code for some map p.

Incompleteness in Turing Categories

Now,

$$\begin{aligned}
0^\bullet p! &= 0^\bullet \langle !p^\bullet, 1\rangle \bullet ! \\
&= 0^\bullet \langle !p^\bullet, 1\rangle f! \\
&= \langle p^\bullet, 0^\bullet\rangle f! \\
&= \langle p^\bullet, 0^\bullet\rangle (\langle 1, !0^{\bullet\bullet}\rangle comp \times 1) e_{pf}!k! \\
&= \langle \langle p^\bullet, 0^{\bullet\bullet}\rangle comp, 0^\bullet\rangle e_{pf}! \\
&= \langle (!0^\bullet p)^\bullet, 0^\bullet\rangle e_{pf}!
\end{aligned}$$

Incompleteness in Turing Categories

If $0^\bullet p! = 0$, then $\langle (0^\bullet p)^\bullet, 0^\bullet \rangle e_{pf} = \langle (0^\bullet p)^\bullet, 0^\bullet \rangle$, and so $\langle (!0^\bullet p)^\bullet, 0^\bullet \rangle ! = 1_1$, which means that $1_1 = 0$.

If $0^\bullet p! = 1_1$, then $p \not\leq 0$ and so $\langle (0^\bullet p)^\bullet, 0^\bullet \rangle e_{pf}!$ cannot be total, which means it must be 0. So again, $0 = 1_1$.

Incompleteness in Turing Categories

... and so there must be predicates of the terminal object that are neither 1_1 nor 0. That is, The setting is non-classical!

$\square$

Logic $\rightarrow$ Turing Category

To connect these two incompleteness results, it would make sense if we could start with a term logic, construct a Turing category using it, and define a provability predicate in that category.

To that end, we introduce the logical system $\mathcal{L}$.

Logic $\to$ Turing Category

Structural inference rules and induction:

$$\frac{\Gamma \vdash \Theta}{\mathcal{D}, \Gamma \vdash \Theta} \qquad \frac{\Gamma \vdash \Theta}{\Gamma \vdash \Theta, \mathcal{D}} \qquad \frac{\mathcal{D}, \mathcal{D}, \Gamma \vdash \Theta}{\mathcal{D}, \Gamma \vdash \Theta} \qquad \frac{\Gamma \vdash \Theta, \mathcal{D}, \mathcal{D}}{\Gamma \vdash \Theta, \mathcal{D}}$$

$$\frac{\Delta, \mathcal{D}, \mathcal{C}, \Gamma \vdash \Theta}{\Delta, \mathcal{C}, \mathcal{D}, \Gamma \vdash \Theta} \qquad \frac{\Gamma \vdash \Theta, \mathcal{C}, \mathcal{D}, \Lambda}{\Gamma \vdash \Theta, \mathcal{D}, \mathcal{C}, \Lambda} \qquad \frac{\Gamma \vdash \Theta, \mathcal{D} \qquad \mathcal{D}, \Delta \vdash \Lambda}{\Gamma, \Delta \vdash \Theta, \Lambda}$$

$$\frac{F(a), \Gamma \vdash \Theta, F(a')}{F(0), \Gamma \vdash \Theta, F(t)}$$

Logic → Turing Category

Logical inference rules:

$$\frac{\Gamma \vdash \Theta, \mathcal{A} \qquad \Gamma \vdash \Theta, \mathcal{B}}{\Gamma \vdash \Theta, \mathcal{A} \wedge \mathcal{B}} \qquad \frac{\mathcal{A}, \Gamma \vdash \Theta}{\mathcal{A} \wedge \mathcal{B}, \Gamma \vdash \Theta} \qquad \frac{\mathcal{B}, \Gamma \vdash \Theta}{\mathcal{A} \wedge \mathcal{B}, \Gamma \vdash \Theta}$$

$$\frac{\mathcal{A}, \Gamma \vdash \Theta \qquad \mathcal{B}, \Gamma \vdash \Theta}{\mathcal{A} \vee \mathcal{B}, \Gamma \vdash \Theta} \qquad \frac{\Gamma \vdash \Theta, \mathcal{A}}{\Gamma \vdash \Theta, \mathcal{A} \vee \mathcal{B}} \qquad \frac{\Gamma \vdash \Theta, \mathcal{B}}{\Gamma \vdash \Theta, \mathcal{A} \vee \mathcal{B}}$$

$$\frac{\Gamma \vdash \Theta, F(t)}{\Gamma \vdash \Theta, \exists x.F(x)} \qquad \frac{F(t), \Gamma \vdash \Theta}{\exists x.F(x), \Gamma \vdash \Theta}$$

Logic → Turing Category

Axioms:

$$\frac{}{x' = 0 \vdash} \qquad \frac{}{\vdash (\exists x. y = x') \vee y = 0} \qquad \frac{}{x' = y' \vdash x = y}$$

$$\frac{}{x = y \vdash P(x) = P(y)} \qquad \frac{}{\vdash x = x} \qquad \frac{}{x = y \vdash y = x}$$

$$\frac{}{x = y, y = z \vdash x = z} \qquad \frac{}{a = b \wedge P(b) \vdash P(a)}$$

Logic $\to$ Turing Category

Define the equivalence relation $\sim$ by $P_1 \sim P_2$ if and only if $P_1 \vdash P_2$ and $P_2 \vdash P_1$. Write the equivalence class of P as $[P]$

We can use this to construct a category $\mathbb{T}_{\mathcal{L}}$:

Objects: $1 = \mathbb{N}^0, \mathbb{N}, \mathbb{N} \times \mathbb{N}, \mathbb{N}^3, \ldots$

Maps: $f : \mathbb{N}^n \to \mathbb{N}^m$ is given by an equivalence class of $n + m$ place predicates, $[P_f]$, such that if $P_f \in [P_f]$ then

$$P_f(x_1, \ldots, x_n, y_1, \ldots, y_m) \wedge P_f(x_1, \ldots, x_n, z_1, \ldots, z_m)$$
$$\vdash y_1 = z_1 \wedge \cdots \wedge y_m = z_m$$

Logic $\to$ Turing Category

Composition:

Given $f : \mathbb{N}^n \to \mathbb{N}^m$, $g : \mathbb{N}^m \to \mathbb{N}^l$,

$$fg :=$$
$$[\exists w_1 \cdots \exists w_m . P_f(x_1, \ldots, x_n, w_1, \ldots, w_m) \wedge P_g(w_1, \ldots, w_m, y_1, \ldots, y_l)]$$

Logic $\to$ Turing Category

Identities:

$1_1 := [true]$

$1_{\mathbb{N}^n} := [x_1 = y_1 \wedge \cdots \wedge x_n = y_n]$

Logic → Turing Category

This forms a category. In fact,

$$\overline{f} := [(\exists \widetilde{w} \,.P_f(\widetilde{x}, \widetilde{w})) \wedge \widetilde{x} = \widetilde{y}]$$

gives that we have a restriction category, and

$$\widehat{f} := [(\exists \widetilde{w} \,.P_f(\widetilde{w}, \widetilde{x})) \wedge \widetilde{x} = \widetilde{y}]$$

gives that we have a range category!

Logic $\to$ Turing Category

Proposition

$\mathbb{T}_{\mathcal{L}}$ is a cartesian restriction category with joins and ranges.

... How exactly we finish the construction and obtain a Turing category is unclear. In particular, how does one define $\bullet$?