The Course So Far:

We've defined various **models of computation** in which we can study what problems can and can't be solved, and what resources the solvable problems require.

The most important model has been the **Turing machine**, which might or might not halt on a given input. **Recursive** sets are decided by always-halting TM's, **recursively enumerable** sets have TM's that accept strings in them and fail to halt on the others. Similarly **total recursive** functions are computed by always-halting TM's, and **partial recursive** function by general TM's.

Some sets and functions are **provably not recursive**:

- The **busy-beaver** function
- HALT $= \{(M, x) : M$ halts on input $x\}$
- $K = \{n : n \in W_n\}$
- any non-trivial property of machines
- Σ^*-CFL

We showed busy-beaver and K to be non-recursive by direct arguments. For the others we used **reductions**: if $A \leq B$ and A is not recursive, then B is not recursive.

A set is **r.e.-complete** if it is r.e and all r.e. languages reduce to it. No r.e.-complete language is recursive. But a language can be non-recursive without being r.e.-complete – $\overline{K}$ is co-r.e.-complete and thus not r.e., and there are languages that are neither r.e. nor co-r.e. at all.

Other Formal Models of Computation:

- **Formal Language Theory:**
  - Regular languages: DFA's, NFA's, or regular expressions
  - Context-free languages: grammars or pushdown automata
- **Boolean:**
  - Compute on boolean values with AND, OR, NOT
  - Boolean expressions compute boolean functions
  - Straight-line boolean programs
  - Fitch proof system: sound and complete
- **First-Order Logic:** ($\exists$, $\forall$) (starting today)
- **Recursive Function Theory:**
  - Primitive Recursive: Bloop computable
  - General Recursive: Floop computable, partial recursive
- **Abstract RAM:**
  - Can simulate TM, vice versa, poly time blowup

In propositional logic we reason about some set of statements, the atomic formulas, each of which is either true or false. **First-order logic with equality (FOL)** is a more complex logical system, in which we deal with the fact that the atomic formulas are *statements about objects*.

To set up an FOL system we need a **domain** D, the set of objects in question, which may be finite or infinite. We then need a **vocabulary**, a formal definition of what atomic statements we may make about the objects. We then will be able to say:

- that two objects are **the same**,
- that an object **exists** with some property, or that
- **all objects** in D have some property.

Formal Definition of a Vocabulary:

A vocabulary Σ is formally made up of three elements:

- The set Φ of **function symbols**, each representing a function from D^k to D for some k. Φ includes the **constant symbols** of the vocabulary, which are thought of as function symbols with $k = 0$.
- The set Π of **predicate symbols**, each representing a **relation** on D, a function from D^k to $\{0, 1\}$ for some k. The **equality sign** is included in Π as a binary relation, written in its usual infix form, e.g. "$s = t$".
- The **arity function** r, which assigns the number of arguments k to each symbol in Φ and Π.

Examples of FOL Vocabularies:

Number Theory: $\Sigma_N = (\Phi_N, \Pi_N, r_N)$

$\Phi_N \quad = \quad \{0,\ \sigma,\ +,\ \times,\ \uparrow\}$

$r_N(0) = 0, r_N(\sigma) = 1, r_N(+) = r_N(\times) = r_N(\uparrow) = 2$

$\Pi_N \quad = \quad \{=,\ <\}, \qquad r_N(=)\ =\ r_N(<)\ =\ 2$

Graph Theory: $\Sigma_G = (\Phi_g, \Pi_g, r_g)$

$\Phi_g \quad = \quad \{s, t\}, \qquad r_g(s)\ =\ r_g(t)\ =\ 0$

$\Pi_g \quad = \quad \{=,\ E\}, \qquad r_g(=)\ =\ r_g(E)\ =\ 2$

Binary String Theory: $\Sigma_W = (\Phi_w, \Pi_w, r_g)$

$\Phi_w \quad = \quad \emptyset$

$\Pi_w \quad = \quad \{=, <, I\} \qquad r_w(=) = r_w(<) = 2, r_w(I) = 1$

Tarski's World: $\Sigma_T = (\Phi_T, \Pi_T, r_T)$

$\Phi_T = \{a, b, c, d, e, f\}$

$\Pi_T = \{$ Tet, Cube, Dodec, Small, Medium, Large, SameSize, SameShape, Larger, Smaller, SameCol, SameRow, Adjoins, LeftOf, RightOf, FrontOf, BackOf, Between $\}$

$$r(a) = r(b) = r(c) = r(d) = r(e) = r(f) = 0$$

$$\begin{aligned} r(\text{Tet}) &= r(\text{Cube}) = r(\text{Dodec}) = r(\text{Small}) \\ &= r(\text{Medium}) = r(\text{Large}) = 1 \end{aligned}$$

$$\begin{aligned} r(\text{SameSize}) &= r(\text{SameShape}) = r(\text{Larger}) = r(\text{Smaller}) \\ &= r(\text{SameCol}) = r(\text{SameRow}) = r(\text{Adjoins}) \\ &= r(\text{LeftOf}) = r(\text{RightOf}) = r(\text{BackOf}) = 2 \end{aligned}$$

$$r(\text{Between}) = 3$$

Inductive Definition of FOL Formulas:

Once we fix a vocabulary Σ we have a set $L(\Sigma)$ of **well-formed formulas**. Entities within formulas have two types, "object" and "boolean". We define valid formulas by induction:

Variables: We have an infinite set

$$V = \{x, y, z, x_1, y_1, z_1, \ldots\}$$

Terms: A **term** is a variable, or a function applied to the correct number of terms. A constant is a special case of the latter.

Formulas: A string is a **well-formed formula** if it is:

- An **atomic formula**, which is a predicate symbol applied to the correct number of terms, or the special atomic formula "$s = t$" where s and t are terms,
- A boolean operator applied to the correct number of formulas, or
- "$\exists x : P$" or "$\forall x : P$" where x is a variable and P is a formula.

Abbreviations:

$$t_1 \leq t_2 \quad \hookrightarrow \quad (t_1 = t_2 \ \vee \ t_1 < t_2)$$

$$1 \quad \hookrightarrow \quad \sigma(0)$$

$$2 \quad \hookrightarrow \quad \sigma(1)$$

$$3 \quad \hookrightarrow \quad \sigma(2)$$

$$t_1 | t_2 \quad \hookrightarrow \quad (\exists x)(t_1 \times x = t_2)$$

$$\text{prime}(t_1) \quad \hookrightarrow \quad 1 < t_1 \ \wedge \ (\forall x)(x | t_1 \rightarrow (x = 1 \vee x = t_1))$$

1. $(\forall x)(x + 0 = x)$
2. $(\exists y)(y + y = x)$
3. $(\forall xy)(x \leq y \ \leftrightarrow \ (\exists z)(x + z = y))$
4. $(\forall x)(\exists y)(x < y \ \wedge \ \text{prime}(y))$
5. $(\forall xy)(\sigma(x) = \sigma(y) \ \rightarrow \ x = y)$
6. $(\forall xy)(x < y \ \rightarrow \ \sigma(x) \leq y)$

1. $(\forall xy)(E(x,y) \rightarrow E(y,x))$

2. $(\forall x)(\neg E(x,x))$

3. $(\forall x)(\exists y)(E(x,y) \vee E(y,x))$

4. $(\forall x)(\neg E(x,s))$

5. $(\exists yz)(y \neq z \wedge E(x,y) \wedge E(x,z))$

6. $(\forall y_1 y_2 y_3)((E(x,y_1) \wedge E(x,y_2) \wedge E(x,y_3))$
$\rightarrow \quad (y_1 = y_2 \vee y_1 = y_3 \vee y_2 = y_3))$

An occurrence of a variable x is *bound* iff it occurs within the scope of a quantifier, $(\forall x)$ or $(\exists x)$. Otherwise the occurrence is *free*.

Examples – Which Variables are Free?

1. $(\exists yz)(y \neq z \wedge E(x,y) \wedge E(x,z))$
2. $(\forall z)(z + x = z)$
3. $(\forall y)(y + x = y)$
4. $(\forall x)(x + x = x)$
5. $x \neq y \ \wedge \ (\exists y)(y < x)$

What Do Variables and Sentences Mean?

Bound variables are dummy variables – you can change their names without affecting the meaning of the formula.

A first-order formula says something *about* its free variables in the context of a particular structure of objects, predicates, and functions. You cannot determine the meaning of the formula without knowing the values of the free variables.

A **sentence** (a formula with no free variables) says something about the entire structure of objects, predicates, and functions. Thus a sentence of Σ_T talks about a particular blocks world, a sentence of Σ_G talks about a particular graph, and Σ_N talks about a particular set of numbers with addition and multiplication defined.

This is true even though there is no **syntactic** reference to the graph G in a Σ_G sentence, or to the string w in a Σ_W sentence. In Σ_G we talk about a vertex as a variable or constant, and an edge as a relation that holds for two vertices. In Σ_W we talk about a position in the input as a variable, and the letter a position contains as unary relation on the position.

A *structure* — also called a *model* — of a vocabulary $\Sigma = (\Phi, \Pi, r)$ is a pair $\mathcal{A} = (U, \mu)$ such that:

$$U = |\mathcal{A}| \neq \emptyset$$

$$\begin{aligned} \mu: V &\to U \\ x &\mapsto x^{\mathcal{A}} \end{aligned}$$

$$\begin{aligned} \mu: \Phi &\to \text{ total functions on } U^{O(1)} \\ \mu: f &\mapsto f^{\mathcal{A}} : U^{r(f)} \to U \end{aligned}$$

$$\begin{aligned} \mu: \Pi &\to \text{ relations on } U^{O(1)} \\ \mu: R &\mapsto R^{\mathcal{A}} \subseteq U^{r(R)} \end{aligned}$$

How's That Again?

In propositional logic we could decide whether an arbitrary compound formula was true or false once we had a **model**, an assignment of a truth value to every atomic formula.

In FOL the model must be more complicated. We need to know what the objects are, and what the relation and function symbols mean. If the **universe** or domain is finite, we can specify this information by finite lookup tables for each function and relation.

To evaluate a formula with free variables, we also need an assignment of an element of the domain to each variable.

FOL Formulas as Propositional Formulas:

Suppose that we know that the domain is finite, and has n elements. We can think of a **sentence** of FOL as implicitly describing a **propositional** formula, where the atomic formulas are particular values of the evaluation function μ. For example:

$$\forall x : A(x) \Leftrightarrow (A(0) \wedge A(1) \wedge \ldots A(n-1))$$

$$\begin{aligned} \exists x : \forall y : B(x,y) \quad &\Leftrightarrow \\ ((B(0,0) \wedge \ldots \wedge B(0,n-1)) \quad &\vee \\ (B(1,0) \wedge \ldots \wedge B(1,n-1)) \quad &\vee \ldots \vee \\ (B(n-1,0) \wedge \ldots \wedge B(n-1,n-1))) & \end{aligned}$$

Treat functions as relations, adding well-definedness formulas.

Proposition 13.1 *Any fixed FOL sentence on a universe of size n is equivalent to a propositional formula of size $n^{O(1)}$.*

Proof: Carry out the above process, noting that the size is only multiplied by $O(n)$ for each quantifier in the formula. ♠

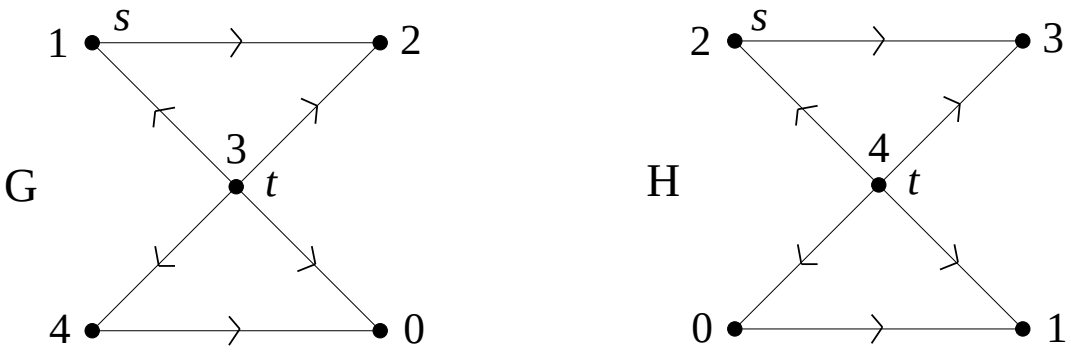
Example: Any world, $\mathcal{W}$, for Tarski's World is is structure of vocabulary Σ_T, i.e, $\mathcal{W} \in \text{STRUC}[\Sigma_T]$.

Example of Graph Structures:

$G = \langle V^G, 1, 3, E^G \rangle \in \text{STRUC}[\Sigma_g]$

$$
\begin{aligned}
V^G &= \{0, 1, 2, 3, 4\} \\
E^G &= \{(1,2), (3,0), (3,1), (3,2), (3,4), (4,0)\}
\end{aligned}
$$

is a structure of vocabulary Σ_g, consisting of a directed graph with two specified vertices s and t. G has five vertices and six edges. (See the figure below which shows G as well as another graph H that is isomorphic but not equal to G.)



Example of a Binary String Structure:

Let w be the string "01101".

$\mathcal{A}_w = \langle \{0, 1, \ldots, 4\}, <, \{1, 2, 4\} \rangle \in \text{STRUC}[\Sigma_s]$

$$\begin{aligned} \Sigma_s &= (\emptyset, \{=, <, S\}, \{\langle =, 2\rangle, \langle <, 2\rangle, \langle S, 1\rangle\}) \\ &= (; <^2, S^1) \end{aligned}$$

1. $(\exists x)(\forall y)(y \leq x \ \wedge \ S(x))$
2. $(\forall xy)((x < y \wedge \neg S(x) \wedge \neg S(y)) \rightarrow (\exists z)(x < z < y))$

$\Sigma_{gen} = (; F^1, P^2, S^2)$

$\mathcal{B}_0 = \langle U_0, F_0, P_0, S_0 \rangle \in \text{STRUC}[\Sigma_{gen}]$

$U_0 = \{\text{Abraham, Isaac, Rebekah, Sarah, } \ldots\}$

$F_0 = \{\text{Sarah, Rebekah, } \ldots\}$

$P_0 = \{\langle \text{Abraham,Isaac} \rangle, \langle \text{Sarah,Isaac} \rangle, \ldots\}$

$S_0 = \{\langle \text{Abraham,Sarah} \rangle, \langle \text{Isaac,Rebekah} \rangle, \ldots\}$

$$\varphi_{sibling}(x, y) \equiv (\exists fm)(x \neq y \wedge f \neq m \wedge P(f, x) \wedge P(f, y) \wedge P(m, x) \wedge P(m, y))$$

$$\varphi_{aunt}(x, y) \equiv (\exists ps(P(p, y) \wedge \varphi_{sibling}(p, s) \wedge (s = x \vee S(x, s))) \wedge F(x)$$

It is perfectly reasonable to have two different models of the same vocabulary, in which different things are true:

$\mathbf{N} = (\mathbf{N}, 0, \sigma, +, \times, \uparrow, <)$, the standard model of the naturals

$\mathbf{Z}/p\mathbf{Z} = (\{0, 1, \ldots, p-1\}, 0, +1_p, +_p, \times_p, \uparrow_p, \emptyset)$, p prime

$\mathbf{N}, \mathbf{Z}/p\mathbf{Z} \in \text{STRUC}[\Sigma_N]$

$$\text{MultInverses} \quad \equiv \quad (\forall u)(u = 0 \vee (\exists v)(u \times v = 1))$$

$$\mathbf{N} \models \neg\text{MultInverses}; \qquad \mathbf{Z}/p\mathbf{Z} \models \text{MultInverses}$$

Beginning to Define Truth:

In propositional logic we inductively defined what it meant for a truth assignment to **satisfy** a formula, or **make it true**. In the same way we can inductively define what it means for a **structure**, of the appropriate vocabulary V, to satisfy a formula.

The first step is to assign an element of the domain to **every term** of $\mathcal{L}(V)$. To do this we inductively **extend the function** μ : terms $\rightarrow |\mathcal{A}|$, (already defined on variables and constants).

$$\mu(f_j(t_1, \ldots, t_{r(f_j)})) \quad = \quad f_j^{\mathcal{A}}(\mu(t_1), \ldots, \mu(t_{r(f_j)}))$$

Now every term has a meaning.

Tarski's Inductive Definition of Truth:

$$
\begin{aligned}
(|\mathcal{A}|, \mu) \models t_1 = t_2 &\Leftrightarrow \mu(t_1) = \mu(t_2) \\
(|\mathcal{A}|, \mu) \models R_j(t_1, \ldots, t_{r(R_j)}) &\Leftrightarrow \langle \mu(t_1), \ldots, \mu(t_{r(R_j)}) \rangle \in R_j^{\mathcal{A}} \\
(|\mathcal{A}|, \mu) \models \neg\varphi &\Leftrightarrow (|\mathcal{A}|, \mu) \not\models \varphi \\
(|\mathcal{A}|, \mu) \models \varphi \vee \psi &\Leftrightarrow (|\mathcal{A}|, \mu) \models \varphi \text{ or } (|\mathcal{A}|, \mu) \models \psi \\
(|\mathcal{A}|, \mu) \models (\forall x)\varphi &\Leftrightarrow (\text{for all } a \in |\mathcal{A}|) \\
& \qquad (|\mathcal{A}|, \mu, a/x) \models \varphi
\end{aligned}
$$

where $\quad (\mu, a/x)(y) = \begin{cases} \mu(y) & \text{if } y \neq x \\ a & \text{if } y = x \end{cases}$

Play Tarski's Truth Game!!!

world: $\mathcal{W}$; sentence: φ; players: A, B

A asserts that $\mathcal{W} \models \varphi$; B denies that $\mathcal{W} \models \varphi$.

The game rules depend inductively on the formula φ:

φ is atomic: A wins iff $\mathcal{W} \models \varphi$.

$\varphi \equiv \alpha \vee \beta$: A asserts $\mathcal{W} \models \alpha$ or A asserts $\mathcal{W} \models \beta$.

$\varphi \equiv \alpha \wedge \beta$: B denies $\mathcal{W} \models \alpha$ or B denies $\mathcal{W} \models \beta$.

$\varphi \equiv \neg\alpha$: A and B switch rôles, and B asserts $\mathcal{W} \models \alpha$.

$\varphi \equiv \exists x(\psi)$: A chooses an element from $|\mathcal{W}|$, assigning it a name n. A asserts that $\mathcal{W}' \models \psi[x \leftarrow n]$.

$\varphi \equiv \forall x(\psi)$: B chooses an element from $|\mathcal{W}|$, assigning it a name n. B denies that $\mathcal{W}' \models \psi[x \leftarrow n]$.

Example: Does $\mathbf{Z}/3\mathbf{Z} \models (\forall u)(u = 0 \vee (\exists v)(u \times v = 1))$?

$$\mathbf{Z}/3\mathbf{Z}, \mu_0 \models (\forall u)(u = 0 \vee (\exists v)(u \times v = 1))$$

$\Leftrightarrow$ (forall $a \in \{0, 1, 2\}$)
$$(\mathbf{Z}/3\mathbf{Z}, \mu_0, a/u) \models (u = 0 \vee (\exists v)(u \times v = 1)$$

$$(\mathbf{Z}/3\mathbf{Z}, \mu_0, 0/u) \models u = 0$$

$\Leftrightarrow$ $(\mu_0, 0/u)(u) = (\mu_0, 0/u)(0)$

$\Leftrightarrow$ $0 = 0$

$$(\mathbf{Z}/3\mathbf{Z}, \mu_0, 1/u) \models (\exists v)(u \times v = 1)$$

$\Leftrightarrow$ (exists $b \in \{0, 1, 2\}$)$(\mathbf{Z}/3\mathbf{Z}, \mu_0, 1/u, b/v) \models (u \times v = 1)$

$$(\mathbf{Z}/3\mathbf{Z}, \mu_0, 1/u, 1/v) \models (u \times v = 1)$$

Similarly,

$$(\mathbf{Z}/3\mathbf{Z}, \mu_0, 2/u) \models (\exists v)(u \times v = 1)$$

[BE] has a clause in the inductive definition for each boolean operator, but it suffices to have only $\wedge$ and $\neg$ as above:

Proposition 13.2

$(|\mathcal{A}|, \mu) \models \varphi \wedge \psi \quad \Leftrightarrow \quad (|\mathcal{A}|, \mu) \models \varphi$ *and* $(|\mathcal{A}|, \mu) \models \psi$

Proof:

$$
\begin{aligned}
& (|\mathcal{A}|, \mu) \models \varphi \wedge \psi \\
\Leftrightarrow\ & (|\mathcal{A}|, \mu) \models \neg(\neg\varphi \vee \neg\psi) \\
\Leftrightarrow\ & \text{not } (|\mathcal{A}|, \mu) \models \neg\varphi \vee \neg\psi \\
\Leftrightarrow\ & \text{not } [((|\mathcal{A}|, \mu) \models \neg\varphi) \text{ or } ((|\mathcal{A}|, \mu) \models \neg\psi)] \\
\Leftrightarrow\ & (|\mathcal{A}|, \mu) \not\models \neg\varphi \text{ and } (|\mathcal{A}|, \mu) \not\models \neg\psi \\
\Leftrightarrow\ & (|\mathcal{A}|, \mu) \models \varphi \text{ and } (|\mathcal{A}|, \mu) \models \psi
\end{aligned}
$$

♠

Similarly defining satisfaction for one quantifier allows you to define it for the other:

Proposition 13.3

$(|\mathcal{A}|, \mu) \models (\exists x)\varphi \quad \Leftrightarrow \quad (\textit{exists } a \in |\mathcal{A}|)(|\mathcal{A}|, \mu, a/x) \models \varphi$

Proof:

$$
\begin{aligned}
& (|\mathcal{A}|, \mu) \models (\exists x)\varphi \\
\Leftrightarrow\ & (|\mathcal{A}|, \mu) \models \neg(\forall x)\neg\varphi \\
\Leftrightarrow\ & (|\mathcal{A}|, \mu) \not\models (\forall x)\neg\varphi \\
\Leftrightarrow\ & \text{ not (for all } a \in |\mathcal{A}|)(|\mathcal{A}|, \mu, a/x) \models \neg\varphi \\
\Leftrightarrow\ & \text{(for some } a \in |\mathcal{A}|)(|\mathcal{A}|, \mu, a/x) \not\models \neg\varphi \\
\Leftrightarrow\ & \text{(for some } a \in |\mathcal{A}|)(|\mathcal{A}|, \mu, a/x) \models \varphi
\end{aligned}
$$

♠

Fitch Proofs for FOL

The Fitch proof system of [BE] can prove FOL formulas as well as propositional ones. We have to add six new proof rules to deal with the new concepts of **identity** and **quantifiers**:

- $=$**-Intro:** Derive $n = n$ (cf. *Atlas Shrugged*?)
- $=$**-Elim:** From $P(n)$ and $n = m$, derive $P(m)$
- $\forall$**-Intro:** (Ordinary form) If for a new variable c you derive $P(c)$, derive $\forall x : P(x)$
- $\forall$**-Intro:** (General conditional form) If from $P(c)$, for a new variable c, you derive $Q(c)$, conclude $\forall x : P(x) \rightarrow Q(x)$
- $\forall$**-Elim:** From $\forall x : S(x)$, derive $S(c)$
- $\exists$**-Intro:** From $S(c)$, derive $\exists x : S(x)$
- $\exists$**-elim:** If from $S(c)$, for a new variable c, you derive Q, then you may derive Q from $\exists x : S(x)$

Coming Attractions:

We will prove Fitch to be **sound** for FOL, following [BE] Section 18.3 with some details on HW#4. The basic idea is very similar to soundness for propositional Fitch. We show by induction on steps of any proof that each statement is true in any **structure** in which all of its premises are true (instead of for any truth assignment).

Then we will prove the **completeness** of Fitch for FOL, following [BE] Chapter 19 with some details on HW#5. The goal is to prove that any FOL-valid sentence can be proved in Fitch. We will do this as follows:

- Define an infinite set of sentences called the **Henkin theory**,
- Show that any propositional extension of the Henkin theory has a model,
- Use propositional completeness to get a propositional Fitch proof of any FOL-valid sentence from the Henkin theory, and finally
- Show that in Fitch we can eliminate every use of the Henkin theory in this proof, to get a Fitch proof of the FOL-valid sentence.