Recall that a machine $\mu = (ID, C, \tilde{\mu})$ contains an identity, communication set, and program. Today we make this fully precise, defining an interactive Turing machine that allows us to play out the dance we saw last week with the execution semantics.

We will assume that you are already familiar with the usual definition of a Turing machine.

Def^n 1

An interactive Turing machine (ITM) μ is a Turing machine with the following extra features:

1) An identity tape containing a description of μ's state transition function and initial tape contents, plus μ's identity (e.g. in $\mathbb{N}$). The entire contents are called the extended identity.

2) An outgoing message tape with a message m and "routing instructions"

3) Externally writeable tapes that are read-only and read-once:
   i) an input tape
   ii) a subroutine-output tape
   iii) a backdoor tape (used to model adversarial influence)

4) A 1-bit activation tape (represents whether μ is active)

5) A read-only, read-once randomness tape (that is "long enough")

6) An external-write instruction (defined later)

7) A read-next-message instruction specifying $t \in \{\text{input, subroutine-output, backdoor}\}$ that jumps to the start of the next message on t (where messages end with a special end-of-message symbol)

And some housekeeping:

Def^n 2

- A configuration of an ITM μ consists of the contents of all tapes, the location of each head within the tape, and the current state. A configuration is active if its activation tape contains 1.
- An activation of an ITM μ is a sequence of configurations representing a computation of μ starting from an active configuration and ending when an inactive one is reached.
- An ITM system is a pair $S = (\mu, C)$ where C is a control function (specifying what classes of messages can be sent).

The control function plays a similar role as the communication sets, but is more general: it can "redirect" messages as well as blocking them. We use this feature to be able to "swap out" machines in an execution.

Def^n 3

An execution of a system $S = (I, C)$ on input z is a sequence of activations, beginning with the initial machine I.

Note that subsequent activations may be for different ITMs as specified in the external-write instruction, which we now define: the outgoing message tape of μ interpreted as a tuple

$$(f, M', t, r, M, m)$$

where

- $f \in \{0, 1\}$ is a forced-write flag
- $M' = (\mu', id')$ is an extended identity
- $t \in \{\text{input}, \text{subroutine-output}, \text{backdoor}\}$
- $r \in \{0, 1\}$ is a reveal-identity flag
- $M = (\mu, id)$ is μ's extended identity
- $m \in \{0, 1\}^*$ is the message.

A control function then maps (f, M', t, r, M, m) to $(\bar{f}, \bar{M'}, \bar{t}, \bar{r}, \bar{M}, \bar{m})$ or disallow.

If $C(f, M', t, r, M, m) =$ disallow, the initial machine I is activated (so μ's activation tape is set to 0 and I's is set to 1).

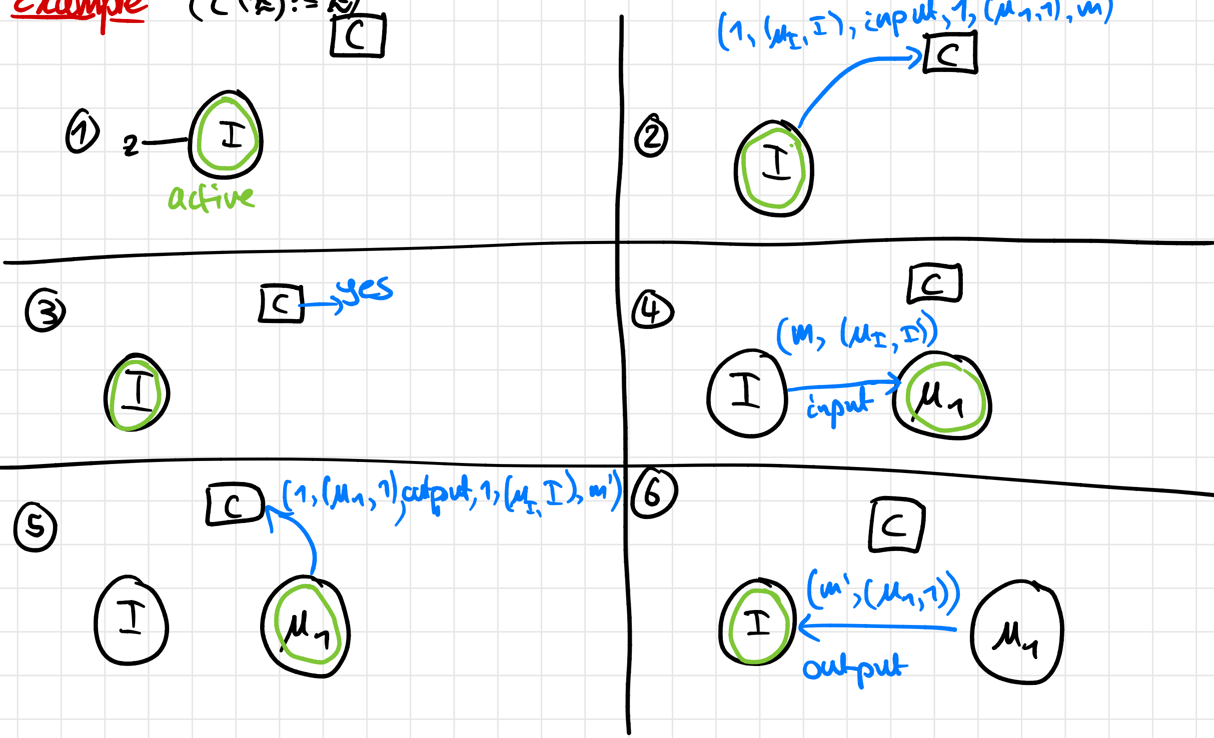
Otherwise:

1) If $f=1$ and an earlier activation has extended identity M', then the message m is written to tape t of M' (at the end of its last activation). If $r=1$, M is also written to t. Then M' is activated.

2) Otherwise, if $f=1$, the next activation will be of M' with its tapes set to their initial configuration, and m appearing at the end of tape t as in case 1). We say M' is invoked.

3) If $f=0$, M' is interpreted in some way as a predicate on extended identities. Take the ITM M'' that satisfies the predicate and was invoked earliest, and deliver the message as in case 1). If no M'' exists, the initial machine is activated.

Example $(C(x) := x)$



We have defined the execution of a single ITM. Next, we define an execution of a protocol.

In order to model parallel sessions, we will interpret an identity as a pair (sid, pid) of the session ID and party ID.

Def^n 4

Let $\pi, \mathcal{A}, \mathcal{E}$ be ITMs (the protocol, adversary, and environment). An execution $EXEC_{\pi,\mathcal{A},\mathcal{E}}(k,z)$ of $\pi, \mathcal{A}, \mathcal{E}$ is an execution of $(\mathcal{E}, C_{exec}^{\pi,\mathcal{A}})$ with initial input z and security parameter k, where $C_{exec}^{\pi,\mathcal{A}}$ is defined as follows:

1) Writes from $\mathcal{E}$: $\mathcal{E}$ can write to any ITMs with the same SID. The control function sets the code of the ITMs to π, and allows $\mathcal{E}$ to set any sender ID. If the PID is $\diamond$, the code is set to $\mathcal{A}$.
The fixed SID $\mathcal{E}$ uses is the test SID.

2) Writes from $\mathcal{A}$: $\mathcal{A}$ is only allowed to write to backdoor tapes, and forced-write must be 0. ($\mathcal{A}$ cannot invoke new machines.)

3) Other writes:
- reveal-sender-id must be set
- if writing to $\mathcal{A}$, must use the backdoor tape, forced-write must be 0, and recipient code must be unspecified.
- if the sender is a main machine, the target tape is subroutine-output, and the recipient does not exist, the value is instead written to $\mathcal{E}$ with both the sender and recipient IDs

The value of $EXEC_{\pi,\mathcal{A},\mathcal{E}}(k,z)$ is the output of $\mathcal{E}$ after an execution.

We now state what it means for one protocol to emulate another.

Def^n 5

A probability distribution ensemble (PDE) is an indexed family of prob. distributions $X = \{X(k,z)\}_{k\in\mathbb{N}, z\in\{0,1\}^*}$. Two PDEs X, Y over

$\{0,1\}^*$ are indistinguishable, written $X \approx Y$, if for all $c, d \in \mathbb{N}$ there exists $k_0 \in \mathbb{N}$ s.t. for all $k > k_0$ and all z of length at most k^d, we have

$$\left| \Pr[X(k,z)=1] - \Pr[Y(k,z)=1] \right| < k^{-c}$$

Intuitively: given inputs of polynomial length, the distinguishability is negligible. We thus define the PDE

$$\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{E}} := \{\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{E}}(k,z)\}_{k \in \mathbb{N}, z \in \{0,1\}^*}$$

Def$^{\underline{n}}$ 6

We say that an ITM π UC-emulates ITM ϕ if for all PPT ITMs $\mathcal{A}$, there exists a PPT ITM S such that for all PPT balanced ITMs $\mathcal{E}$:

$$\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{E}} \approx \mathrm{EXEC}_{\phi,S,\mathcal{E}}$$

Two crucial pieces remain:

1) How do we define a PPT ITM? (See also "balanced" above)
2) How do we state and prove a composition theorem for Def$^{\underline{n}}$ 6?

Note that simply arguing $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{E}} \approx \mathrm{EXEC}_{\phi,S,\mathcal{E}} \approx \mathrm{EXEC}_{\psi,T,\mathcal{E}}$ and using transitivity does not answer 2), because we want to be able to "swap out subroutines".

We conclude this week by addressing 1).

Polynomial time

For $T: \mathbb{N} \to \mathbb{N}$, a Turing machine μ is T-bounded if, on an input of length n, it halts after at most $T(n)$ steps. Simply extending our ITM model by requiring each activation to be T-bounded does not suffice: two machines could "play ping-pong", activating each other with increasingly-long messages (assuming $T(n) > n$) and using an unbounded amount of resources.

Instead, we require each message to contain an **import** value. A machine's **budget** n in a given configuration is the sum of received imports minus the sent imports. An ITM is then T-bounded if the number of steps since invocation is at most $T(n)$.

Def^n 7

Let $T: \mathbb{N} \to \mathbb{N}$. An ITM μ is **locally T-bounded** if at all prefixes of executions of systems of ITMs, all ITMs M with program μ have taken at most $T(n)$ steps where n is the sum of imports on M's externally writeable tapes minus that on its outgoing message tape. Then

1) A locally T-bounded ITM that never sets forced-write $= 1$ is **T-bounded**.
2) A locally T-bounded ITM whose external writes are to only T-bounded ITMs is T-bounded.

An ITM μ is **PPT** if there exists a polynomial p such that μ is p-bounded.

To justify this definition, we show a PPT ITM can be simulated by a polynomially-bounded standard TM.

Theorem

Let $T: \mathbb{N} \to \mathbb{N}$ be an increasing function s.t. for all $n, m \in \mathbb{N}$

$$T(n+m) \geq T(n) + T(m) \quad \text{(T super-additive)}$$

Given a system (I, C), if I is T-bounded and C can be computed in time $t(\cdot)$,* then an execution of (I, C) can be simulated on a TM in $O(T(n)\, t(T(n)))$.

* We require that C never increases messages' import for protocol executions.

Proof

I claim the number of configurations in an execution of (I, C) is at most $T(n)$, where n is the initial import of I.

Recall an execution is a sequence of ITM configurations. Let M_i be the set of ITMs active before the i^{th} configuration, and for each $\mu \in M_i$ let $n_{\mu,i}$ be its total budget immediately before the i^{th} configuration.

Since I is T-bounded, for all $\mu \in M_i$, μ is also T-bounded and thus

$$
\begin{aligned}
i = \sum_{\mu \in M_i} (\text{steps taken by } \mu) &\leq \sum_{\mu \in M_i} T(n_{\mu,i}) \\
&\leq T\Big(\sum_{\mu \in M_i} n_{\mu,i}\Big) \\
&\leq T(n)
\end{aligned}
$$

To simulate the execution, a TM μ writes all of the configurations of (I, C) on its tape and accepts if the halting configuration of I accepts.

The overall amount of time spent by μ is $T(n)$ (since $i \leq T(n)$ and T is super-additive) plus the time spent evaluating C (for each of the $i \leq T(n)$ configurations). Since C is evaluated at most $T(n)$ times, the bound follows. $\square$

The final modification is in Def^n 4: we require that messages sent from π to A have zero import (since they are modelling artifacts).