On to real programming languages...

Base types

Up to now, we've formulated "base types" (e.g. `Nat`) by adding them to the syntax of types, extending the syntax of terms with associated constants (`zero`) and operators (`succ`, etc.) and adding appropriate typing and evaluation rules. We can do this for as many base types as we like.

For more theoretical discussions (as opposed to programming) we can often ignore the term-level inhabitants of base types, and just treat these types as uninterpreted constants.
E.g., suppose `B` and `C` are some base types. Then we can ask (without knowing anything more about `B` or `C`) whether there are any types `S` and `T` such that the term

```
(λf:S. λg:T. f g) (λx:B. x)
```

is well typed.

The `Unit` type

`t`	::=	...	*terms*
		`unit`	*constant* `unit`
`v`	::=	...	*values*
		`unit`	*constant* $\mathtt{unit}$
`T`	::=	...	*types*
		`Unit`	*unit type*

New typing rules

$$\boxed{\Gamma \vdash \mathtt{t} : \mathtt{T}}$$

$$\Gamma \vdash \mathtt{unit} : \mathtt{Unit} \qquad \text{(T-Unit)}$$

Sequencing

$\mathtt{t}$	::=	...	*terms*
		$\mathtt{t}_1\,;\mathtt{t}_2$	

Sequencing

$$\begin{array}{lll} \mathtt{t} & ::= & \ldots \qquad\qquad\qquad\qquad\qquad\qquad \textit{terms} \\ & & \mathtt{t}_1\mathtt{;}\mathtt{t}_2 \end{array}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{t}_1\mathtt{;}\mathtt{t}_2 \longrightarrow \mathtt{t}_1'\mathtt{;}\mathtt{t}_2} \qquad \text{(E-Seq)}$$

$$\mathtt{unit;t}_2 \longrightarrow \mathtt{t}_2 \qquad \text{(E-SeqNext)}$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{Unit} \qquad \Gamma \vdash \mathtt{t}_2 : \mathtt{T}_2}{\Gamma \vdash \mathtt{t}_1\mathtt{;}\mathtt{t}_2 : \mathtt{T}_2} \qquad \text{(T-Seq)}$$

Derived forms

- Syntatic sugar
- Internal language vs. external (surface) language

Sequencing as a derived form

$$\mathtt{t}_1\mathtt{;}\mathtt{t}_2 \quad \stackrel{\text{def}}{=} \quad (\lambda \mathtt{x}\mathtt{:}\mathtt{Unit}.\mathtt{t}_2)\ \mathtt{t}_1$$
$$\text{where } \mathtt{x} \notin \mathit{FV}(\mathtt{t}_2)$$

Equivalence of the two definitions

[board]