Here’s where I throw in a bunch of LaTeX that I have been using. For sanity’s sake I need your help to grow this organically. Ideally it will help you with typsetting your own materials. Let me know what pieces are missing and I will add them.

1 Explicit Sets, Set Comprehensions, and Tuples

I use the `braket.sty` package to typeset tuples and sets:

\[
\{ a, b, c \} \\
\{ a \in A \mid a \notin B \} \\
\langle a, b, c \rangle
\]

2 Object Language Bits

Why so blue? Well, because I typeset object language items using the `\mbsf{}` or `\tbsf{}` macros defined in `defs.tex`.

3 Typesetting BNF’s

Here’s the language of Boolean and Arithmetic Expressions:

\[
t \in \text{TERM}, \quad v \in \text{VALUE}, \quad nv \in \text{NUM} \\
t ::= \quad \text{true} \mid \text{false} \mid \text{if} \ t \ \text{then} \ t \ \text{else} \ t \\
\mid \quad z \mid \text{succ}(t) \mid \text{pred}(t) \mid \text{zero?}(t) \\
v ::= \quad \text{true} \mid \text{false} \mid nv \\
nv ::= \quad z \mid \text{succ}(nv)
\]

If you look at the file, you will see things like `<if> t <then> t <else> t` which come out like `if t then t else t`. I am using the keyword feature of the `semantic.sty` package to define special keywords that I can refer to in angle brackets. In this particular case, the definition in `defs.tex` that brings things to life is:

```
\reservestyle{\oblang}{\tbsf}
\oblang{if-then-else, 
    true, false, if[if\}], then[\;then\;], else[\;else\;]}
```

*© Ronald Garcia. Not to be copied, used, or revised without explicit written permission from the copyright owner.
The \texttt{reservestyle} macro creates a new “style” command (in this case \texttt{oblang}) which I can use to create keywords (the thing before the square brackets) and what text they will render (the thing inside the square brackets) using the style function given in the \texttt{reservestyle} keyword. For example, \texttt{\texttt{<if}>} actually translates to \texttt{\tbsf{if\;}} where \texttt{\;} is the latex command for “leave some space”.

4 A Function Definition

Let \texttt{nat : Num \to N} be defined by

\[
\begin{align*}
\text{nat}(z) &= 0 \\
\text{nat}(\text{succ}(nv)) &= 1 + \text{nat}(nv).
\end{align*}
\]

5 An Inductive Definition

\[
\begin{array}{c}
\text{true} \Downarrow \text{true} \quad \text{(etru)} \\
\text{false} \Downarrow \text{false} \quad \text{(efalse)} \\
\text{if} \; t_1 \; \text{then} \; t_2 \; \text{else} \; t_3 \Downarrow v_2 \quad \text{(eif-t)} \\
\text{if} \; t_1 \; \text{then} \; t_2 \; \text{else} \; t_3 \Downarrow v_3 \quad \text{(eif-f)}
\end{array}
\]

\[
\text{z} \Downarrow \text{z} \quad \text{(ez)} \\
\text{succ}(t) \Downarrow \text{succ}(nv) \quad \text{(esucc)} \\
\text{pred}(t) \Downarrow nv \quad \text{(epred)}
\]

\[
\begin{array}{c}
\text{t} \Downarrow z \quad \text{(ezero?-z)} \\
\text{zero?}(t) \Downarrow \text{true} \quad \text{(ezero?-s)}
\end{array}
\]

In the past I have tended to use \texttt{\textbackslash{inference}} from semantic.sty to define inductive rules, and \texttt{\textbackslash{infer}} from proof.sty to write derivations, because I think the \texttt{\textbackslash{inference}} is prettier, but using it to write derivations is a big space hog. I’m moving toward consistently using \texttt{\textbackslash{infer}} for everything just to keep it simple. Hence, the rules above are typeset using \texttt{\textbackslash{infer}}.

Beware! \texttt{\textbackslash{inference}} and \texttt{\textbackslash{infer}} take their arguments in opposite order, so if you switch, then you have to switch the arguments too, otherwise you’ll end up with errors or upside-down derivations/rules.

6 A Concrete Derivation

As an example derivation of an entailment relation, here is a derivation of \(\emptyset \vdash A \lor \bot \Rightarrow A \text{ true} \), meaning that it is a theorem of constructive propositional logic (it can be entailed with no assumptions):

\[
\begin{array}{c}
\text{\{ } A \lor \bot \text{ \} } \vdash A \lor \bot \quad \text{(hyp)} \\
\text{\{ } A \lor \bot, A \text{ \} } \vdash A \text{ true} \quad \text{(hyp)} \\
\text{\{ } A \lor \bot \text{ \} } \vdash A \text{ true} \quad \text{(hyp)} \\
\text{\{ } A \lor \bot, \bot \text{ \} } \vdash A \text{ true} \quad \text{(hyp)} \\
\emptyset \vdash A \lor \bot \Rightarrow A \text{ true} \quad \text{(\Rightarrow I)}
\end{array}
\]

7 Abstract Derivations and Properties of Derivations

This is useful for seeing how to typeset abstract derivations, where you don’t see everything.

Let \(P(D)\) be a predicate on (or property of) derivations \(D\). Then \(P\) holds for all derivations \(D\) if:

1. \(P \left( \text{true} \in \text{TERM} \ (\text{r-true}) \right) \) holds;
2. $P \left( \text{false} \in \text{TERM} \right) \ (\text{r-false})$ holds;

3. If $P \left( r_1 \in \text{TERM} \right)$, $P \left( r_2 \in \text{TERM} \right)$, and $P \left( r_3 \in \text{TERM} \right)$ hold then

$$P \left( r_1 \in \text{TERM} \quad r_2 \in \text{TERM} \quad r_3 \in \text{TERM} \right) \ (\text{r-if})$$

holds.

Note the use of \deduce instead of infer to omit the horizontal bar. The \sarray macro, profided by defs.tex is a trick to get the large parentheses to size correctly (try removing it and see what happens).

Notice that when writing properties of derivations, I wrap the derivation in an \sarray form, which is defined in defs.tex. That keeps there from being extra annoying blank space at the bottom of the derivation. I don’t have a good explanation for why this is necessary I’m afraid.

8 A Definition by Cases

This definition uses the case environment to lay out the three possible cases. Define $eval_{BA} : \text{TERM} \rightarrow \{ \text{true}, \text{false} \} \cup \mathbb{N}$ by

$$eval_{BA}(t) = \begin{cases} 
\text{true} & \text{if } t \downarrow \text{true} \\
\text{false} & \text{if } t \downarrow \text{false} \\
\text{nat}(nv) & \text{if } t \downarrow nv 
\end{cases}$$

9 A Proposition

Proposition 1 (Inversion).

1. If $\text{true} \downarrow v$ then $v = \text{true}$.

2. If $\text{false} \downarrow v$ then $v = \text{false}$.

3. If $t_1 \text{ if } t_2 \text{ else } t_3 \downarrow v$ then either
   (a) $t_1 \downarrow \text{true}$ and $t_2 \downarrow v$ or
   (b) $t_1 \downarrow \text{false}$ and $t_3 \downarrow v$

4. If $z \downarrow v$ then $v = \text{z}$.

5. If $\text{succ}(t) \downarrow v$ then $t \downarrow v_1$, $v_1 \in \text{NUM}$, and $v = \text{succ}(v_1)$.

6. If $\text{pred}(t) \downarrow v$ then $t \downarrow \text{succ}(v)$ and $v \in \text{NUM}$.

7. If $\text{zero?}(t) \downarrow v$ then either
   (a) $t \downarrow \text{z}$ and $v = \text{true}$ or
   (b) $t \downarrow \text{succ}(nv)$ and $v = \text{false}$.

10 A Logical Statement (about derivations)

To prove these propositions, we first expand them to be formal statements about derivations. For example, item 7 expands to the following:

**Proposition 2.** $\forall D. D :: \text{zero?}(t) \downarrow v \Rightarrow ((\exists \mathcal{E} :: t \downarrow \text{z}) \land v = \text{true}) \lor ((\exists \mathcal{E} :: t \downarrow \text{succ}(nv)) \land v = \text{false})$