Lifting Boolean Algebras to Function Spaces

**Theorem 1** Let $A$ be a non-empty set and let $B$ be a complete atomic Boolean algebra. Then the operations on $B$ can be lifted on the function space, $A \rightarrow B$ by point-wise definition, and $A \rightarrow B$ forms a complete atomic Boolean algebra.

By point-wise definition, we mean that we define operations on functions in terms of the operations on the values of the functions. For example $f \land g = h$ iff for all $a$ in $A$, $h(a) = f(a) \land g(a)$. We can write these using lambda notation as follows:

1. $f \land_{(A,B)} g = \lambda a. f(a) \land_B g(a)$
2. $f \lor_{(A,B)} g = \lambda a. f(a) \lor_B g(a)$
3. $\neg_{(A,B)} f = \lambda a. \neg_B f(a)$
4. $0_{(A,B)} = \lambda a. 0_B$
5. $1_{(A,B)} = \lambda a. 1_B$

It is easy to see that the resulting structure is a complete Boolean algebra. The only tricky part is showing that it is atomic.

The atoms are not the constant functions giving the same atom. Instead they are functions that are 0 for all but a single element of the domain, and the image of that element is an atom in $B$. I.e., for all $a \in A$ and $b \in B$, define $f_{a,b} : A \rightarrow B$ s.t. $f_{a,b}(a) = b$ and $f_{a,b}(x) = 0$ for all $x \neq a$.

It is easy to show that each of these elements are atoms. That is, there is nothing between them and 0, and all functions in $A \rightarrow B$ have a lower bound that is an atom.

Now we define $BOOL$ to be the smallest set such that

1. $t \in BOOL$
2. If $a \in TYPE$ and $b \in BOOL$ then $(a,b) \in BOOL$.

By the above theorem, every element of $BOOL$ is a complete atomic Boolean algebra.

In the particular where the function space is $(A,t)$, each element of the function space can be considered to be the characteristic function of a relation on $A$.

**Generalizing negation, conjunction, and disjunction**

We can use the fact that we can lift the Boolean algebra operations to higher types in order to define $\neg$, $\land$, and $\lor$ on functional types.

1. $\neg_t$, $\land_t$, and $\lor_t$ are the standard logical operations on truth values.
2. Let $a = (b,c)$ be a type in $BOOL$. Then $\neg_a$, $\land_a$, and $\lor_a$ are obtained from the similar operations on $c$ by lifting. I.e., If $P, Q \in a$, and $x \in b$, then
   - $\neg_a = \lambda P. \lambda x. \neg_c P(x)$.
   - $\land_a = \lambda P. \lambda Q. \lambda x. P(x) \land_c Q(x)$.
Recall that the type of the semantics of intransitive verbs like “walk” is \(\langle e, t \rangle\). We typically write the translation as \(\lambda s. \text{walk}(s)\). By the above definition, the translation of “not walking” should be:

\[
\neg_{\langle e, t \rangle} \lambda s. \text{walk}(s).
\]

Therefore the translation of “John is not walking” is

\[
(\lambda v. v@j)(\lambda x. \neg_t \text{walk}(x)) = (\lambda x. \neg_t \text{walk}(x))@j = \neg_t \text{walk}(j)
\]

Similar examples show the same behavior for transitive verbs, which have type \(\langle \langle \langle e, t \rangle, t \rangle, \langle e, t \rangle \rangle\).

Here is the translation of “not pet”:

\[
\neg_{\langle \langle \langle e, t \rangle, t \rangle, \langle e, t \rangle \rangle} \lambda o. \lambda s. \neg_t o@((\lambda x. \text{pet}(s, x))) = \lambda o. \lambda s. \neg_t o@((\lambda x. \text{pet}(s, x)) = \lambda s. \neg_t ((\lambda x. \text{pet}(s, x)))@r = \lambda s. \neg_t \text{pet}(s, r)
\]

To translate “Ann does not pet Rover”, we begin with “not pet Rover”:

\[
(\lambda o. \lambda s. \neg_t (o@((\lambda x. \text{pet}(s, x))))) (\lambda v. v@r) = \lambda s. \neg_t ((\lambda v. v@r)@((\lambda x. \text{pet}(s, x))) = \lambda s. \neg_t ((\lambda x. \text{pet}(s, x))@r = \lambda s. \neg_t \text{pet}(s, r)
\]

We omit the details, but “Ann does not pets Rover” then translates as \(\neg_t \text{pet}(a, r)\) as expected.

[Note that “does” can simply be ignored as a syntactic place-holder.]

The same kind of construction works with “and”. For example, we can translate “Ann and every man” as

\[
(\lambda v. v@a) \land_{\langle (e, t), t \rangle} (\lambda v. \forall x. (\text{man}(x) \rightarrow v@x))
\]

which is equivalent to

\[
\lambda v. (v@a \land \forall x. (\text{man}(x) \rightarrow v@x))
\]

As expected, “Ann and every man walk” is translated by applying the above representation to the meaning of walking, obtaining:

\[
\text{walk}(a) \land \forall x. (\text{man}(x) \rightarrow \text{walk}(x))
\]