1. INTRODUCTION

We show how the HOL Light primitive inference rules imply the truth tables for the Boolean functions $\neg$, $\land$, $\lor$, and $\Rightarrow$. We give a mathematical idealization of HOL Light using alpha-equivalence classes of terms. It would be possible to rewrite HOL Light to do this, using DeBruijn trees, but then the printer would not be able to print our preferred bound variables. Using alpha-equivalence classes of terms requires us to change some HOL Light primitive inference rules, replacing BETA with BETA_CONV, and removing the alpha-equivalence aspects of TRANS, EQ_MP and DEDUCT_ANTISYM_RULE.

We explain HOL Light with informal mathematical proofs about the set of theorems. To justify this, note that the HOL4 Logic manual explains the meaning of HOL4 in terms of a model of ZFC. We can think of HOL Light as describing actual sets and functions, rather than thinking of HOL Light as a formalization of set theory. It would be interesting to formalize our proofs below in HOL Light, and at the end we report some progress towards this. It would be also interesting to compare the HOL Light Boolean basics to Andrews’s equality version of Church’s theory [1, Ch. 5]. Note that the equals sign =, naturally used in our mathematical explanations, is also the main symbol in HOL Light, and it’s important to keep the two uses of = straight.

This note was stimulated by questions asked by Vladimir Voevodsky, who has been developing homotopy type theory [5]. Thanks to Mark Adams, Rob Arthan and Konrad Slind for some helpful conversations. Tom Hales’s compact description of HOL Light [4] was a good resource.

2. THE INFECTION RULES AND BOOLEAN DEFINITIONS

The set TYPE of types is freely generated by variables and type constants, which can be specified by the user (with new_type). Initially we have only three type constants, the types bool and ind, and the type function $\rightarrow$, so for two types $\alpha$ and $\beta$, we have the type $\alpha \rightarrow \beta$. 
We have a set of terms \( \text{TERM} \) which is defined inductively as a disjoint union of variables, constants, lambda abstractions, and function applications by the following rules. Every term \( t \) has a type, say \( \alpha \), and this is indicated by the type annotation \( t : \alpha \). A variable is an identifier together with a type \( x : \alpha \). Constants can be defined by the user (with \text{new\_constant} in HOL Light). Initially we have only the polymorphic constant \( = \) of type \( A \to A \to \text{bool} \). Given a variable \( x \) of type \( \alpha \) and term \( t \) of type \( \beta \), we have the lambda abstractions \( \lambda x. t \) of type \( \alpha \to \beta \). Given a term \( f \) of type \( \alpha \to \beta \) and a term \( a \) of type \( \alpha \), we have the function application \( f a \). So we can describe our inductively defined set \( \text{TERM} \) in BNF form as on p 15 of the HOL4 manual Logic

\[
\text{VAR} ::= \langle \text{ident} \rangle : \text{type} \\
\text{TERM} ::= \text{VAR} \mid \text{CONSTANT} \mid \lambda \text{VAR.} \text{TERM} \mid \text{TERM TERM}
\]

We need to take into account the readable HOL Light expressions which the parser systematically replaces with elements of \( \text{TERM} \). The parser replaces \( \forall x. t \) with \( (\forall)(\lambda x. t) \), \( p \land q \) with \( (\land) p q \), \( (\equiv) : \text{bool} \to \text{bool} \to \text{bool} \) with \( (=) : \text{bool} \to \text{bool} \to \text{bool} \).

So we define the set \( \tilde{T} \) of representatives of readable terms by

\[
\begin{align*}
\text{INFIX} & ::= \equiv | \Rightarrow | \lor | \land | = \quad \text{BINDER} ::= \forall | \exists \\
\tilde{T} & ::= \text{TERM} | \lambda \text{VAR.} \tilde{T} | \tilde{T} \tilde{T} | \tilde{T} \text{INFIX} \tilde{T} | \text{BINDER VAR.} \tilde{T}
\end{align*}
\]

By HOL Light definitions, for any \( i \in \text{INFIX} \), \( (i) \in \text{TERM} \). Furthermore for any \( b \in \text{BINDER} \), \( (b) \in \text{TERM} \). Our definitions of \( (i) \) and \( (b) \) are given in \text{FORALL\_DEF} through \text{EXISTS\_DEF} below.

Now we define the equivalence relation on the set \( \tilde{T} \) generated by

\[
\begin{align*}
&x \ i \ y \sim (i) \ x \ y, \quad \text{for} \ x, y \in \tilde{T} \text{ and } i \in \text{INFIX}, \\
&b \ x. \ t \sim (b) \ (\lambda x. t), \quad \text{for} \ x \in \text{VAR}, \ t \in \tilde{T}, \text{ and } b \in \text{BINDER}.
\end{align*}
\]

Call \( \mathcal{T} \) the set of equivalence classes of \( \tilde{T} \) under this equivalence relation. We will indicate equivalence classes with \( = \), and write e.g.

\[
\forall x. \ t = (\forall)(\lambda x. t) \in \mathcal{T}, \quad p \land q = (\land) p q \in \mathcal{T}.
\]

It is clear that the inclusion \( \text{TERM} \to \tilde{T} \) gives a bijection \( \text{TERM} \to \mathcal{T} \). We can therefore replace \( \text{TERM} \) with \( \mathcal{T} \), which contains the readable expressions which are equal to their parser replacements in \( \text{TERM} \).

We will use a typed version of the theorem [2] that beta-conversion is well defined modulo alpha-equivalence classes. Consider the set \( \mathcal{T}/\text{ALPHA} \) of alpha-equivalence classes of \( \mathcal{T} \). We use Barendregt’s convention that \text{term} means an element of \( \mathcal{T}/\text{ALPHA} \). We will also use Barendregt’s convention of not indicating equivalence classes, and will write, for two alpha-equivalent elements of \( \mathcal{T} \),

\[
(x. f) = (y. f [y/x]) \in \mathcal{T}/\text{ALPHA}, \text{ instead of writing } \\
(x. f) \sim (y. f [y/x]) \text{ or } [(x. f)] = [(y. f [y/x])].
\]
We sketch a proof of Barendregt’s theorem [2], as he does not explain this in his article on the typed lambda calculus [3]. The set TERM can be thought of as a set of trees where the symbol $\lambda$ is a node with two children, a variable and a tree. Consider the set MD-TERM of modified Debruijn trees mD-trees, where

(1) a $\lambda$ node has only one child, an mD-tree, and

(2) a variable can be a positive integer $n$ if there is a $\lambda$ node $n$ levels in the mD-tree above it, and

(3) an integer variable in the mD-tree under a $\lambda$ node counts the number of levels up the tree of its binding lambda abstraction.

The only variables occurring in a mD-tree are free variables. MD-TERM is not defined in BNF form, but it is clear that there is a bijection $\text{TERM}/\text{ALPHA} \rightarrow \text{MD-TERM}$ which replaces each bound variable by the number of levels up the tree of its binding lambda abstraction. We define easily beta-conversion $(\lambda . t) s \mapsto t[s]$ in MD-TERM: $t[s]$ is the mD-tree obtained by replacing each positive integer in $t$ pointing to our top $\lambda$ with the mD-tree $s$. This proves that beta-conversion is well defined in $\text{TERM}/\text{ALPHA}$.

HOL Light has the feature of type inference, by which the types needed for terms can be defined by context. The only way this arises here is with the polymorphic constant $=$, which is defined to have type $A \rightarrow A \rightarrow \text{bool}$. $A$ is a type variable which can be assigned the value of any type. Given two terms $l$ and $r$ of the same type, the term $l = r$ is constructed as follows. Let $\alpha$ be the type of $l$ and $r$. We perform the type variable instantiation $[\alpha/A]$ in $=$ to produce the term

$l =: \alpha \rightarrow \alpha \rightarrow \text{bool} r$. With the exception of proof of Theorem SYM below, we will not write the polymorphic type annotation $\alpha \rightarrow \alpha \rightarrow \text{bool}$, and instead write $l = r$.

We define a sequent to be an ordered pair $(A, t)$ where $A$ is a finite set of terms of type bool, and $t$ is a term of type bool. We define SEQUENT to be the set of sequents.

SEQUENT has a subset THEOREM, whose elements are called theorems. THEOREM is defined inductively by some basic structure (the types bool and ind, the constants $=$ and $\rightarrow$), the 10 HOL Light primitive inference rule functions REFL, TRANS, MK_COMB, ABS, BETA_CONV, ASSUME, EQ_MP, DEDUCT_ANTISYM_RULE, INST and INST_TYPE, and 3 axioms (Extensionality, Infinity and Choice). Actually we have a version of THEOREM given by any collection of extra structure given by using new_definition, new_type, new_constant, and new_axiom. The computationally defined notion of theorem creates, by the axioms of ZFC or any comparable foundations, a well-defined subset THEOREM of SEQUENT called the deductive closure.
If the sequent \((A, t)\) is a theorem, then we will write
\[ A \vdash t, \]
meaning the sentence “\(t\) is deduced from the set of assumptions \(A\)”,
with \(\vdash\) being the verb, in the same way that \(6 + 7 = 13\) is a sentence,
with \(=\) being the verb. This seems to be the style of the HOL4 manual
Description and and Andrew’s treatments of Equality in [1, Ch. 5].

We’ll call axioms all the “basic” mathematical theorem based on re-
sults that HOL Light doesn’t prove, whether these theorems are created
by the primitive inference rules, new axiom or new definition, on the
grounds that it’s enough of a challenge to distinguish between the two
uses of the word theorem, elements of THEOREM, and mathematical
theorems proved about the set THEOREM.

We’ll now list the informal mathematical theorems which follow from
these deductive rules. The first nine primitive inference rules yield

**Axiom REFL.** For any term \(t\),
\[ \vdash t = t. \]

**Axiom TRANS.** Take any terms \(t_1, t_2\) and \(t_3\), and assume
\[ A1 \vdash t_1 = t_2 \quad A2 \vdash t_2 = t_3. \] Then
\[ A1 \cup A2 \vdash t_1 = t_3. \]

**Axiom MK_COMB.** Take any terms \(f, g, x\) and \(y\), and assume
\[ A1 \vdash f = g \quad A2 \vdash x = y. \] Then
\[ A1 \cup A2 \vdash f \, x = g \, y. \]

**Axiom ABS.** Assume that
\[ A \vdash t_1 = t_2. \]
Then for any variable \(x\) which is not free in \(A\),
\[ A \vdash (\lambda x. \, t_1) = (\lambda x. \, t_2). \]

**Axiom BETA_CONV.** For any variable \(v\) and terms \(bod\) and \(arg\),
\[ \vdash (\lambda v. \, bod) \, arg = bod[arg/v]. \]

**Axiom ASSUME.** For any term \(t\),
\[ \{ t \} \vdash t. \]

**Axiom EQ_MP.** Take any terms \(t_1\) and \(t_2\), and assume
\[ A1 \vdash t_1 \leftrightarrow t_2 \quad A2 \vdash t_1. \] Then
\[ A1 \cup A2 \vdash t_2. \]

**Axiom DEDUCT_ANTISYM_RULE.** Assume that
\[ A \vdash p \quad B \vdash q. \] Then
\[ (A - \{ q \}) \cup (B - \{ p \}) \vdash p \leftrightarrow q. \]

Following the HOL Light reference manual, the minus sign ‘-’ denotes
set-theoretic difference, so \(A - \{ q \} = \{ x \in A \mid x \neq q \}\). Hales [4] uses
instead a slash ‘/’ for set-theoretic difference.
Axiom **INST_TYPE**. Assume $A \vdash t$.

For any list of type variables $tv_1, \ldots, tv_n$ and any list of types $ty_1, \ldots, ty_n$,

$A[ty_1, \ldots, ty_n/tv_1, \ldots, tv_n] \vdash t[ty_1, \ldots, ty_n/tv_1, \ldots, tv_n]$.

We obtain this theorem by replacing all instances of each type variable $tv_i$ by the corresponding type $ty_i$ in both assumptions and conclusions.

Take any term $t$ and any list of variables $x_1, \ldots, x_n$ and terms $t_1, \ldots, t_n$ such that $x_i$ and $t_i$ have the same type. Then

$t[t_1, \ldots, t_n/x_1, \ldots, x_n]$ means the (equivalence class of a) term obtained by replacing the free occurrences of each $x_i$ in $t$ with $t_i$, performing alpha-equivalences on bound variables to avoid variable capture errors. There are many ways to perform these alpha-equivalences, but Barendregt [2] proves that the alpha-equivalence class $t[t_1, \ldots, t_n/x_1, \ldots, x_n]$ is well-defined, and only depends on the alpha-equivalence class $t$. Now we state the last axiom given by the primitive inference rules.

**Axiom INST.** Assume that $A \vdash t$. Take any list of variables $x_1, \ldots, x_n$ and terms $t_1, \ldots, t_n$ such that $x_i$ and $t_i$ have the same type. Then

$A[t_1, \ldots, t_n/x_1, \ldots, x_n] \vdash t[t_1, \ldots, t_n/x_1, \ldots, x_n]$.

The next basic theorems we'll look at are given by the occurrences of new_basic_definition in bool.ml defining the logical constants T, F, $\forall$, $\land$, $\Rightarrow$, $\neg$, $\lor$ and $\exists$.

**Axiom T_DEF.** $\vdash T = ((\lambda p: \text{bool}. p) = (\lambda p: \text{bool}. p))$

**Axiom F_DEF.** $\vdash F = \forall p: \text{bool}. p$

**Axiom FORALL_DEF.** $\vdash (\forall) = \lambda P: A \rightarrow \text{bool}. \ P = \lambda x. \ T$.

**Axiom AND_DEF.**

$\vdash (\land) = \lambda p. q. (\lambda f: \text{bool} \rightarrow \text{bool} \rightarrow \text{bool}. \ f \ p \ q) = (\lambda f. \ f \ T \ T)$.

**Axiom IMP_DEF.** $\vdash (\Rightarrow) = \lambda p. q. \ p \land q \leftrightarrow p$.

**Axiom NOT_DEF.** $\vdash (\neg) = \lambda p. \ p \Rightarrow F$.

**Axiom OR_DEF.**

$\vdash (\lor) = \lambda p. q. \forall r. (p \Rightarrow r) \Rightarrow (q \Rightarrow r) \Rightarrow r$.

**Axiom EXISTS_DEF.**

$\vdash (\exists) = \lambda P: A \rightarrow \text{bool}. \forall q. (\forall x. \ P \ x \Rightarrow q) \Rightarrow q$.

3. **Theorems derived from equal.ml and bool.ml**

From the code of AP_TERM, AP_THM, SYM, and in equal.ml we easily glean simple proofs of the three theorems
**Theorem AP_TERM.** Take terms f, x and y, and assume 
A ⊢ x = y. Then 
A ⊢ f x = f y.

*Proof.* By Axiom REFL,
A ⊢ f = f. Apply Axiom MK_COMB to this theorem and our assumption theorem, noting that A ∪ ∅ = A. □

**Theorem AP_THM.** Take terms f, g and x, and assume 
A ⊢ f = g. Then 
A ⊢ f x = g x.

*Proof.* By Axiom REFL,
A ⊢ x = x. Apply Axiom MK_COMB as in the proof above. □

**Theorem SYM.** Take terms l and r of the same type, and assume 
A ⊢ l = r. Then 
A ⊢ r = l.

*Proof.* Let α be the type of l, which is also the type of r. Apply Theorem AP_TERM with (=):α→α→bool and our assumption theorem. Then 
A ⊢ (=) l = (=) r. Apply Theorem AP_THM to this theorem and the term l. Then 
A ⊢ (=) l l = (=) r l. Thus 
A ⊢ (l = l) = (r = l). By Axiom REFL 
A ⊢ l = l. Apply Axiom EQ_MP and use A ∪ ∅ = A. □

Now we prove more useful versions of the Boolean axioms defining the functions ¬, ∧, ∨, and ⇒.

**Theorem AND_THM.** Let t1 and t2 be any terms of type bool, and let f′ be a variable of type bool→bool→bool that is not free in either t1 or t2. Then 
A ⊢ t1 ∧ t2 = ((λf′. f′ t1 t2) ⇔ (λf. f T T)).

*Proof.* Take distinct variables p′, q′ of type bool that are not free in either t1 or t2. Then 
(λp q. (λf:bool→bool→bool. f p q) = λf. f T T) = 
(λp′ q′. (λf′:f′ p′ q′) = λf′. f T T),
since terms are elements of Ξ/ALPHA. By Axiom AND_DEF,
A ⊢ (λf. f′ p′ q′) = λf′. f′ T T. By Theorem AP_TERM,
A ⊢ (t1 = (λp′ q′. (λf′:f′ p′ q′) = λf′. f′ T T) t1). 
By Axioms BETA_CONV and TRANS,
A ⊢ (t1 = λq′. (λf′:f′ t1 q′) = λf′. f′ T T.
Use the same argument with t2. □

Similarly we have ⇒, ∨ and ¬ theorems.
Theorem IMP_THM. Let \( t_1 \) and \( t_2 \) be any terms of type bool. Then
\[
\vdash (t_1 \Rightarrow t_2) = (t_1 \land t_2 \Leftrightarrow t_1)
\]

Theorem OR_THM. Let \( t_1 \) and \( t_2 \) be any terms of type bool, and let \( r' \) be a variable of type bool that is not free in either \( t_1 \) or \( t_2 \). Then
\[
\vdash t_1 \lor t_2 = \forall r'. (t_1 \Rightarrow r') \Rightarrow (t_2 \Rightarrow r') \Rightarrow r'.
\]

Theorem NOT_THM. Let \( t \) be any term of type bool. Then
\[
\vdash \neg t = (t \Rightarrow F).
\]

Now we prove the first results of bool.ml.

Theorem TRUTH. \( \vdash T \).

Proof. By Axiom T_DEF and Theorem SYM,
\[
\vdash ((\lambda p:bool. p) = (\lambda p:bool. p)) = T. \text{ By Axiom REFL,}
\]
\[
\vdash (\lambda p:bool. p) = (\lambda p:bool. p). \text{ Apply Axiom EQ_MP.} \quad \square
\]

Our next result follows immediately from Theorem SYM, Theorem TRUTH, Axiom EQ_MP, and \( A \cup \emptyset = A \).

Theorem EQT_ELIM. Take any term \( tm \), and assume
\( A \vdash tm \Leftrightarrow T \). Then
\( A \vdash tm \).

The converse takes a little more work.

Theorem EQT_INTRO. Take any term \( tm \) and assume that
\( A \vdash tm \Leftrightarrow T \).

Proof. By Theorem TRUTH,
\( \vdash T \). Apply Axiom DEDUCT_ANTISYM_RULE to our assumption theorem and this theorem. Then
\( A - \{T\} \vdash tm \Leftrightarrow T \). We have two cases. The case \( T \not\in A \) follows from
\( A - \{T\} = A \).

Next suppose that \( T \in A \). By Axiom ASSUME,
\( \{T\} \vdash T \). Apply Axiom DEDUCT_ANTISYM_RULE to the theorem from last paragraph and this theorem. Then
\( A \vdash (tm \Leftrightarrow T) \Leftrightarrow T \), since \( ((A - \{T\}) - \{T\}) \cup \{T\} = A \), since \( T \in A \).
Apply Theorem EQT_ELIM. \quad \square

Theorem CONJ. Take two terms \( t_1 \) and \( t_2 \), and assume
\( A_1 \vdash t_1 \quad \quad A_2 \vdash t_2 \).
Then
\( A_1 \cup A_2 \vdash t_1 \land t_2 \).

Proof. By our assumptions and Theorem EQT_INTRO,
\( A_1 \vdash t_1 \Leftrightarrow T \quad \quad A_2 \vdash t_2 \Leftrightarrow T \).
Choose \( f':\text{bool} \rightarrow \text{bool} \rightarrow \text{bool} \) which is not free in \( t_1, t_2, \) A1 or A2. Apply Theorem AP_TERM with \( f' \) to the first theorem. Then \( A_1 \vdash f' \ t_1 \iff f' \ T. \) Applying Axiom MK_COMB to this and the previous theorem, 
\[
A_1 \cup A_2 \vdash f' \ t_1 t_2 \iff f' \ T \ T.
\]
By Theorem AND_THM, 
\[
\vdash t_1 \land t_2 = ((\lambda f'. \ f' \ t_1 t_2) \iff \lambda f'. \ f' \ T \ T). \text{ Apply Theorem SYM, Axiom EQ_MP and the theorem of the last paragraph.} \]

The next two results give something of a converse of Theorem CONJ, and rest on the well-known lambda calculus facts that an ordered pair \((x, y)\) can be implemented as the abstraction \((\lambda f. \ f \ x \ y)\), and applying an ordered pair to \((\lambda p \ q. \ p)\) (resp. \((\lambda p \ q. \ q)\)) gives the projection on the first (resp. 2nd) coordinate.

**Theorem CONJUNCT1.** Given terms \( l \) and \( r \) of type bool, assume 
\[
A \vdash l \land r.
\]
Then 
\[
A \vdash l.
\]

*Proof.\* Choose a variable \( f' \) of type \( \text{bool} \rightarrow \text{bool} \rightarrow \text{bool} \) that is not free in either \( l \) or \( r \). By Theorem AND_THM, 
\[
\vdash l \land r = ((\lambda f'. \ f' \ l \ r) = \lambda f'. \ f' \ T \ T). \text{ Apply Axiom EQ_MP to this theorem and our assumption theorem. Then}
\]
\[
A \vdash (\lambda f'. \ f' \ l \ r) = (\lambda f'. \ f' \ T \ T). \text{ Choose distinct free variables } x \text{ and } y \text{ of type bool that are not free in } l \text{ or } r. \text{ Apply Theorem AP_THM to this theorem equation and the term } (\lambda x \ y. \ x). \text{ Then}
\]
\[
A \vdash (\lambda f'. \ f' \ l \ r) (\lambda x \ y. \ x) = (\lambda f'. \ f' \ T \ T) (\lambda x \ y. \ x).
\]
Apply Axiom BETA_CONV, Axiom TRANS and Theorem SYM a number of times. Then 
\[
A \vdash l = T. \text{ Apply Theorem EQT_ELIM.} \]

By the same argument using \((\lambda p \ q. \ q)\) instead \((\lambda p \ q. \ p)\), we have

**Theorem CONJUNCT2.** Given terms \( l \) and \( r \) of type bool, assume 
\[
A \vdash l \land r.
\]
Then 
\[
A \vdash r.
\]

**Theorem MP.** Take two terms \( t_1 \) and \( t_2 \), and assume 
\[
A_1 \vdash t_1 \Rightarrow t_2 \quad \text{A2} \vdash t_1. \text{ Then}
\]
\[
A_1 \cup A_2 \vdash t_2.
\]

*Proof.\* By IMP_THM, 
\[
\vdash (t_1 \Rightarrow t_2) = (t_1 \land t_2 \iff t_1). \text{ By our first assumption theorem, Axiom EQ_MP and } A_1 \cup \emptyset = A_1,
\]
A1 ⊢ t1 ∧ t2 ⇔ t1. By our second assumption theorem and Theorem EQT_INTRO,
A2 ⊢ t1 ⇔ T. Apply Axiom TRANS to these two theorems. Then
A1 ∪ A2 ⊢ t1 ∧ t2 ⇔ T. By Theorem EQT_ELIM,
A1 ∪ A2 ⊢ t1 ∧ t2. Apply Theorem CONJUNCT2. □

**Theorem DISCH.** Take any term u and assume
A ⊢ t. Then
A - {u} ⊢ u ⇒ t.

*Proof.* By Theorems IMP_THM and SYM,
⊢ (u ∧ t ⇔ u) = (u ⇒ t).
By Axiom ASSUME,
{u} ⊢ u {u ∧ t} ⊢ u ∧ t.
Apply Theorem CONJ to our assumption theorem and the first theorem, and apply Theorem CONJUNCT1 to the second. Then
A ∪ {u} ⊢ u ∧ t {u ∧ t} ⊢ u.
Applying Axiom DEDUCT_ANTISYM_RULE to these theorems,
A - {u} ⊢ u ∧ t ⇔ u.
Apply Axiom EQ_MP to the theorems of the last two paragraphs. □

**Theorem UNDISCH.** Given terms t1 and t2 of type bool, assume
A ⊢ t1 ⇒ t2. Then
A ∪ {t1} ⊢ t2.

*Proof.* By Axiom ASSUME,
{t1} ⊢ t1.
Apply Theorem MP to the assumption theorem and this theorem. □

Here’s a basic fact about the finite set of assumptions in a theorem.

**Theorem ExpandAssumption.** Given a theorem
A ⊢ t, let B be a finite set of terms of type bool with A ⊂ B. Then
B ⊢ t.

*Proof.* We prove this by induction on the size n of the set B - A. In the base case when n = 0, B = A and so
B ⊢ t.
Assume that n ≥ 0 and the result is true for n. Assume that B - A has size n + 1. Choose u ∈ B - A, and let C = B - {u}. Thus B is the disjoint union of C and {u}, A ⊂ C and C - A has size n. Then by our inductive hypothesis,
C ⊢ t. By Theorem DISCH,
C ⊢ u ⇒ t, since u /∈ C. Apply Theorem UNDISCH. □
Theorem IMP\_ANTISYM\_RULE. Take any two terms $t_1$ and $t_2$ of type bool, and assume
$A_1 \vdash t_1 \Rightarrow t_2$ \quad $A_2 \vdash t_2 \Rightarrow t_1$. Then
$$(A_1 - \{t_1\}) \cup (A_2 - \{t_2\}) \vdash t_1 \Leftrightarrow t_2.$$

Proof. Apply Theorem UNDISCH to our assumption theorems. Then
$A_2 \cup \{t_2\} \vdash t_1$ \quad $A_1 \cup \{t_1\} \vdash t_2$.
Apply Axiom DEDUCT\_ANTISYM\_RULE. \hfill \Box$

Combining the last two results, we have

Theorem IMP\_ANTISYM\_RULE\_EZ. Take any two terms $t_1$ and $t_2$ of type bool, and assume
$A_1 \vdash t_1 \Rightarrow t_2$ \quad $A_2 \vdash t_2 \Rightarrow t_1$. Then
$A_1 \cup A_2 \vdash t_1 \Leftrightarrow t_2$.

Theorem EQ\_IMP\_RULE. Take any terms $t_1$ and $t_2$ of type bool, and assume
$A \vdash t_1 \Leftrightarrow t_2$. Then
$A \vdash t_1 \Rightarrow t_2 \quad A \vdash t_2 \Rightarrow t_1$.

Proof. By Axiom ASSUME,
$\{t_1 \Leftrightarrow t_2\} \vdash t_1 \Leftrightarrow t_2$ \quad $\{t_1\} \vdash t_1$. By Axiom EQ\_MP,
$\{t_1, t_1 \Leftrightarrow t_2\} \vdash t_2$. Applying Theorem DISCH twice,
$\vdash (t_1 \Leftrightarrow t_2) \Rightarrow (t_1 \Rightarrow t_2)$. Apply Theorem MP to this theorem and
our assumption theorem. Then
$A \vdash t_1 \Rightarrow t_2$. This proves the first part our result.
By Theorem SYM and our assumption theorem,
$A \vdash t_2 \Leftrightarrow t_1$. Apply the first part our result to this theorem. \hfill \Box$

Theorem IMP\_TRANS. Take any terms $t_1$, $t_2$, and $t_3$ of type bool, and assume
$A_1 \vdash t_1 \Rightarrow t_2$ \quad $A_2 \vdash t_2 \Rightarrow t_3$. Then
$A_1 \cup A_2 \vdash t_1 \Rightarrow t_3$.

Proof. We follow the proof of Theorem EQ\_IMP\_RULE. By Axioms
ASSUME and EQ\_MP, we see that
$\{t_1, t_1 \Rightarrow t_2, t_2 \Rightarrow t_3\} \vdash t_3$.
Apply Theorem DISCH three times to obtain
$\vdash (t_2 \Rightarrow t_3) \Rightarrow (t_1 \Rightarrow t_2) \Rightarrow (t_1 \Rightarrow t_3)$. Apply Theorem MP twice
using our assumption theorems. \hfill \Box$

Theorem SPEC. Given a term $t$ of type bool and a variable $x$, assume
$A \vdash \forall x. \ t$. Assume the variable $u$ has the same type as $x$, and that $x$
is not free in $A$. Then
$A \vdash t[u/x]$. 

Proof. Let $\alpha$ be the type of $x$. Then $(\lambda x. t)$ has type $\alpha \rightarrow \text{bool}$. Apply Axiom INST_TYPE to Axiom FORALL_DEF and the type substitution $[\alpha/A]$. Then

$\vdash (\forall) = (\lambda P: \alpha \rightarrow \text{bool}. \ P = \lambda x. \ T)$.

Apply Theorem AP_THM and the term $(\lambda x. t)$. Then

$\vdash (\forall x. \ t) = (\lambda P: \alpha \rightarrow \text{bool}. \ P = \lambda x. \ T)(\lambda x. \ t)$.

By Axiom EQ_MP and our assumption theorem, $\vdash (\lambda P: \alpha \rightarrow \text{bool}.\ P = \lambda x. \ T)(\lambda x. \ t)$. By Axiom BETA_CONV,

$\vdash (\lambda x. \ t) = (\lambda x. \ T)$. Apply Theorem AP_THM to this theorem and the term $x$. Then

$\vdash (\lambda x. \ t) x = (\lambda x. \ T) x$. Apply Axioms BETA_CONV and TRANS twice and Theorem SYM once. Then

$\vdash t = T$. Apply Axiom INST to this theorem and the term substitution $[u/x]$. Then

$\vdash t[u/x] = T$, since $x$ is not free in $A$, and $T$ has no free variables.

Apply Theorem EQT_ELIM. □

Theorem GEN. Assume

$A \vdash t$. Assume that the variable $x$ is not free in $A$. Then

$A \vdash \forall x. \ t$.

Proof. As we showed in the proof of Theorem SPEC,

$\vdash (\forall x. \ t) = (\lambda P: \alpha \rightarrow \text{bool}. \ P = \lambda x. \ T)(\lambda x. \ t)$.

where $\alpha$ is the type of $x$. By Axiom BETA_CONV and Theorem SYM,

$\vdash ((\lambda x. \ t) = \lambda x. \ T) = \forall x. \ t$.

By Theorem EQT_INTRO and our assumption theorem,

$A \vdash t \iff T$. Apply Axiom ABS. Then

$A \vdash (\lambda x. \ t) \iff \lambda x. \ T$. Apply Axiom EQ_MP and the theorem of the last paragraph. □

Theorem EXISTS. Let $p$ be any term of type bool. Take any term $u$ and any variable $x$ where $u$ and $x$ have the same type. Assume

$A \vdash p[u/x]$. Then

$A \vdash \exists x. \ p$.

Proof. By Axiom EXISTS_DEF,

$\vdash (\exists) = \lambda P:A \rightarrow \text{bool}. \ \forall q. \ (\forall x. \ P x \Rightarrow q) \Rightarrow q$.

Apply Theorem AP_THM to this theorem and the term $P$. Then

$\vdash (\exists) P = (\lambda P:A \rightarrow \text{bool}. \ \forall q. \ (\forall x. \ P x \Rightarrow q) \Rightarrow q) P$.

By Axiom BETA_CONV, Axiom TRANS and Theorem SYM,

$\vdash (\forall q. \ (\forall x. \ P x \Rightarrow q) \Rightarrow q) = (\exists) P$.

By Axiom ASSUME, Theorem SPEC and the substitution $[x/x]$,

$\{\forall x. \ P x \Rightarrow q\} \vdash P x \Rightarrow q$.

By Theorem UNDISCH and then Theorem DISCH,
\{P \ x\} \vdash (\forall x. P \ x \Rightarrow q) \Rightarrow q. By Theorem GEN,
\{P \ x\} \vdash \forall q. (\forall x. P \ x \Rightarrow q) \Rightarrow q.

Apply Axiom EQ_MP to the theorem of the last paragraph. Then
\{P \ x\} \vdash (\exists) P. By Theorem DISCH,
\vdash P \ x \Rightarrow (\exists) P. Let \alpha be the type of x. Then (\lambda x. p) has type \alpha \rightarrow \text{bool}.

Apply Axiom INST_TYPE and the type substitution [\alpha/A]. Then
\vdash (P: \alpha \rightarrow \text{bool}) x: \alpha \Rightarrow (\exists: \alpha \rightarrow \text{bool} \rightarrow \text{bool}) P.

Apply Axiom INST and the variable instantiation [(\lambda x. p)/P]. Then
\vdash (\lambda x. p) \ x \Rightarrow \exists x. p.

Apply Axiom INST and the variable instantiation [u/x]. Then
\vdash (\lambda x. p) \ u \Rightarrow \exists x. p.

By Axiom BETA_CONV and Theorem SYM,
\vdash p[u/x] = (\lambda x. p) \ u. By Axiom EQ_MP and the assumption theorem,
A \vdash (\lambda x. p) \ u.

Apply Theorem MP and the theorem of the last paragraph. □

**Theorem DISJ1.** Given terms t1 and t2 of type bool, assume
A \vdash t1. Then
A \vdash t1 \lor t2.

*Proof.* By Theorems OR_THM and SYM,
\vdash (\forall r. (p \Rightarrow r) \Rightarrow (q \Rightarrow r) \Rightarrow r) = p \lor q.

By Axiom ASSUME and Theorem MP,
\{p, p \Rightarrow r\} \vdash r. By Theorem DISCH applied to q \Rightarrow r and p \Rightarrow r,
\{p\} \vdash (p \Rightarrow r) \Rightarrow (q \Rightarrow r) \Rightarrow r. By Theorem GEN applied to r,
\{p\} \vdash \forall r. (p \Rightarrow r) \Rightarrow (q \Rightarrow r) \Rightarrow r.

Apply Axiom EQ_MP to the theorem of the last paragraph. Then
\{p\} \vdash p \lor q. By Theorem DISCH,
\vdash p \Rightarrow p \lor q.

By Axiom INST applied to the variable instantiation [t1, t2/p, q],
\vdash t1 \Rightarrow t1 \lor t2. Apply Theorem MP and our assumption theorem. □

The proof of the next result is similar.

**Theorem DISJ2.** Given terms t1 and t2 of type bool, assume
A \vdash t2. Then
A \vdash t1 \lor t2.

*Proof.* Again by Theorems OR_THM and SYM,
\vdash (\forall r. (p \Rightarrow r) \Rightarrow (q \Rightarrow r) \Rightarrow r) = p \lor q.

By Axiom ASSUME and Theorem MP,
\{q, q \Rightarrow r\} \vdash r.

By the same argument as in the proof of Theorem DISJ1,
\{q\} \vdash p \lor q and then
\vdash t2 \Rightarrow t1 \lor t2. Apply Theorem MP to our assumption theorem. □
Theorem DISJ_CASES. Take three terms t1, t2 and t of type bool, and assume
A ⊢ t1 ∨ t2 A1 ⊢ t A2 ⊢ t. Then
A ∪ (A1 - {t1}) ∪ (A2 - {t2}) ⊢ t.

Proof. Apply Theorem DISCH to our second and third assumption theorems. Then
A1- {t1} ⊢ t1 ⇒ t A2 - {t2} ⊢ t2 ⇒ t.

By Theorem OR_THM,
⊢ t1 ∨ t2 = ∀r'. (t1 ⇒ r') ⇒ (t2 ⇒ r') ⇒ r', where r' is a variable of type bool that is not free in either t1, t2 or A. By Axiom EQ_MP,
A ⊢ ∀r'. (t1 ⇒ r') ⇒ (t2 ⇒ r') ⇒ r'.

By Theorem SPEC applied to t,
A ⊢ (t1 ⇒ t) ⇒ (t2 ⇒ t) ⇒ t.

Apply Theorem MP to the two theorems of the first paragraph. □

Theorem SIMPLE_DISJ_CASES. Take three terms p, q and r of type bool, and assume
A ∪ {p} ⊢ r B ∪ {q} ⊢ r.

Then
(A - {p}) ∪ (B - {q}) ∪ {p ∨ q} ⊢ r.

Proof. By Axiom ASSUME,
{p ∨ q} ⊢ p ∨ q. Apply Theorem DISJ_CASES. □

Theorem NOT_ELIM. Let t be a term of type bool. Assume
A ⊢ ¬t. Then
A ⊢ t ⇒ F.

Proof. By Theorem NOT_THM,
⊢ ¬t = (t ⇒ F). Apply Axiom EQ_MP. □

Theorem NOT_INTRO. Let t be a term of type bool, and assume
A ⊢ t ⇒ F. Then
A ⊢ ¬t.

Proof. By the above argument and Theorem SYM,
⊢ (t ⇒ F) = ¬t. Apply Axiom EQ_MP. □

Theorem EQF_INTRO. Let t be a term of type bool, and assume
A ⊢ ¬t. Then
A ⊢ t ⇔ F.

Proof. By Axiom F_DEF,
⊢ F = ∀p. p, where p has type bool. By Theorem UNDISCH,
{F} ⊢ ∀p. p.

By Theorem SPEC applied to this theorem and the substitution [t/p],
\{F\} ⊢ t. By Theorem DISCH,
\vdash F \Rightarrow t. By our assumption and Theorem NOT_ELIM,
A ⊢ t ⇒ F. Apply Theorem IMP_ANTISYM_RULE_EZ and the previous theorem.

\textbf{Theorem \textsc{Eqf}_elim.} Let \( t \) be a term of type bool, and assume
\[ A \vdash t \iff F. \]
Then
\[ A \vdash \neg t. \]

\textit{Proof.} By Theorem EQ_IMP_RULE,
\[ A \vdash t \Rightarrow F. \]
Apply Theorem NOT_INTRO.

\textbf{Theorem \textsc{Contr.}} Let \( t \) be a term of type bool, and assume
\[ A \vdash F. \]
Then
\[ A \vdash t. \]

\textit{Proof.} By Axiom \textsc{F_def},
\[ \vdash F = \forall p. \ p. \]
By Theorem EQ_IMP_RULE,
\[ \vdash F \Rightarrow \forall p. \ p. \]
By Theorem UNDISCH,
\[ \{F\} \vdash \forall p. \ p. \]
By Theorem SPEC applied to the variable instantiation \([t/p]\),
\[ \{F\} \vdash t. \]
By Theorem DISCH,
\[ \vdash F \Rightarrow t. \]
Apply Theorem MP and our assumption theorem.

\section{Lemmas needed for the Boolean truth tables}

Theorems \textsc{Truth} and DISCH immediately imply

\textbf{Theorem \textsc{TermImpliesT}.} For any term \( t \) of type bool,
\[ \vdash t \Rightarrow T. \]

Now we have the “dual” result.

\textbf{Theorem \textsc{FimpliesTerm}.} For any term \( t \) of type bool,
\[ \vdash F \Rightarrow t. \]

\textit{Proof.} By Axiom \textsc{Assume},
\[ \{F\} \vdash F. \]
By Theorem \textsc{Contr},
\[ \{F\} \vdash t. \]
Apply Theorem DISCH.

\textbf{Theorem \textsc{T_elim}.} For any term \( t \) of type bool, assume
\[ A \vdash T \Rightarrow t. \]
Then
\[ A \vdash t. \]

\textit{Proof.} By Theorem \textsc{TermImpliesT},
\[ \vdash t \Rightarrow T. \]
By Theorem IMP_ANTISYM_RULE_EZ applied to this theorem and our assumption theorem,
\[ A \vdash t \iff T. \]
Apply Theorem EQT_ELIM.
By Theorems NOT_INTRO and EQF_INTRO, we have

**Theorem F_ELIM.** For any term t of type bool, assume
A ⊢ t ⇒ F. Then
A ⊢ t ⇔ F.

Now we have versions of Theorems CONJUNCT1 and CONJUNCT2.

**Theorem Conjunct12.** For any terms l and r of type bool,

=⇒ l ∧ r ⇒ l

Proof. By Axiom ASSUME,
{l ∧ r} ⊢ l ∧ r. By Theorems CONJUNCT1 and CONJUNCT2,
{l ∧ r} ⊢ l
{l ∧ r} ⊢ r. Apply Theorem DISCH. □

**Theorem HalfDoubleNegation.** For any term p of type bool,

p ⇒ ¬(¬p).

Proof. By Axiom ASSUME,
{¬p} ⊢ ¬p. By Theorem NOT_ELIM,
{¬p} ⊢ p ⇒ F. By Theorems UNDISCH and DISCH,
{p} ⊢ ¬p ⇒ F. By Theorem NOT_INTRO,
{p} ⊢ ¬(¬p). Apply Theorem DISCH. □

5. Truth Tables

Now we turn to the truth table results for the Boolean functions ¬, ∧, ∨ and ⇒, which in HOL Light are computed by ITAUT_TAC.

**Theorem AndTruthTable.**

=⇒ T ∧ T ⇔ T

=⇒ F ∧ T ⇔ F

=⇒ T ∧ F ⇔ F

=⇒ F ∧ F ⇔ F.

Proof. Theorems TRUTH, CONJ and EQT_INTRO immediately imply the first theorem

=⇒ T ∧ T ⇔ T. By Theorem Conjunct12,
=⇒ F ∧ T ⇒ F. By Theorem Conjunct12,
=⇒ T ∧ F ⇒ F. By Theorem Conjunct12,
=⇒ F ∧ F ⇒ F. By Theorem CONJUNCT2, we have our last three theorems. □

**Theorem OrTruthTable.**

=⇒ T ∨ T ⇔ T

=⇒ F ∨ T ⇔ T

=⇒ T ∨ F ⇔ T

=⇒ F ∨ F ⇔ F.

Proof. Theorems DISJ1, DISJ2 and EQT_INTRO imply the first three theorems. By Axiom ASSUME,
{F} ⊢ F. By Theorem SIMPLE_DISJ_CASES with A = B = ∅,
{F ∨ F} ⊢ F. By Theorem DISCH,
=⇒ F ∨ F ⇒ F. By Theorem F_ELIM, we have our fourth theorem. □
Theorem NotTruthTable.
\[ \vdash \neg F \iff T \quad \vdash \neg T \iff F. \]

Proof. By Axiom REFL and Theorem EQ_IMP_RULE,
\[ \vdash F \Rightarrow F. \]
Apply Theorems NOT_INTRO and EQT_INTRO for our first theorem.
Apply Theorem HalfDoubleNegation to T. Then
\[ \vdash T \Rightarrow \neg(\neg T). \]
By Theorem T_ELIM,
\[ \vdash \neg(\neg T). \]
Apply Theorem EQF_INTRO. \[ \square \]

Theorem ImpTruthTable.
\[ \vdash (T \Rightarrow T) \iff T \quad \vdash (F \Rightarrow T) \iff T \]
\[ \vdash (F \Rightarrow F) \iff T \quad \vdash (T \Rightarrow F) \iff F. \]

Proof. Theorems TermImpliesT, FimpliesTerm and EQT_INTRO imply the first three theorems. By Theorem NotTruthTable,
\[ \vdash \neg T \iff F. \]
By Theorems NOT_THM (using the term T) and SYM,
\[ \vdash (T \Rightarrow F) = \neg T. \]
By Axiom TRANS applied to the last two theorem,
\[ \vdash (T \Rightarrow F) \iff F. \]
\[ \square \]

6. Extra functorial lemmas

Now we prove some “functorial” properties of the Boolean functions.

Theorem ConjImpFunctoriality. Take any terms l, r, x and y of type bool, and assume
\[ A \vdash l \Rightarrow x \quad A \vdash r \Rightarrow y. \]
Then
\[ A \vdash l \land r \Rightarrow x \land y. \]

Proof. By Theorem Conjunct12,
\[ \vdash l \land r \Rightarrow l \quad \vdash l \land r \Rightarrow r. \]
By Theorem IMP_TRANS and our assumption theorems,
\[ A \vdash l \land r \Rightarrow x \quad A \vdash l \land r \Rightarrow y. \]
Apply Theorem CONJ. \[ \square \]

Theorem ConjIffFunctoriality. Take any terms l, r, x and y of type bool, and assume
\[ A \vdash l \iff x \quad A \vdash r \iff y. \]
Then
\[ A \vdash l \land r \iff x \land y. \]

Proof. By Theorems EQ_IMP_RULE and ConjImpFunctoriality
\[ A \vdash l \land r \Rightarrow x \land y \quad A \vdash x \land y \Rightarrow l \land r. \]
Apply Theorem IMP_ANTISYM_RULE_EZ. \[ \square \]

Theorem ImpInvariance. For any terms l, r, x and y of type bool, assume
\[ A \vdash l \iff x \quad A \vdash r \iff y. \]
Then
\[ A \vdash (l \Rightarrow r) \iff (x \Rightarrow y). \]
Proof. Axiom ASSUME and Theorem EQ_IMP_RULE applied to the assumption theorems yields three theorems

\[ A \vdash x \Rightarrow l \quad \{ l \Rightarrow r \} \vdash l \Rightarrow r \quad A \vdash r \Rightarrow y. \]

Apply Theorem IMP_TRANS twice. Then

\( A \cup \{ l \Rightarrow r \} \vdash x \Rightarrow y. \) By Theorems DISCH and ExpandAssumption, \( A \vdash (l \Rightarrow r) \Rightarrow (x \Rightarrow y). \) Similarly we prove
\( A \vdash (x \Rightarrow y) \Rightarrow (l \Rightarrow r). \)

Apply Theorem IMP_ANTISYM_RULE_EZ. \( \square \)

7. Towards an HOL Light formalization

We’ve formalized two of the bool.ml proofs above:

```
let EQT_INTRO th =
    let th1 = DEDUCT_ANTISYM_RULE (ASSUME (concl th)) TRUTH in
    let th2 = EQT_ELIM(ASSUME(concl th1)) in
    let pth = DEDUCT_ANTISYM_RULE th2 th1 in
    EQ_MP pth th;;

let EQF_ELIM th =
    try (NOT_INTRO o fst o EQ_IMP_RULE) th
    with Failure -> failwith "EQF_ELIM";;
```

It might be difficult to show for certain that these functions are identical to the HOL Light functions. But they appear to work, as we built a new hol_light directory with this code substituted into bool.ml, and this ran with now errors in the new directory:

```
ocaml
#use "hol.ml";;
needs "RichterHilbertAxiomGeometry/HilbertAxiom_read.ml";;
needs "RichterHilbertAxiomGeometry/UniversalPropCartProd.ml";;
```

References