A *theorem* in a logic $L$ is a statement that is provable in $L$. An example of such a statement is “There are infinitely many prime numbers.” Associated with a logic is a *theorem proving procedure* $P_L$ that tries to find a proof of a statement $X$ by generating all possible proofs of $X$. If $X$ is provable, $P_L$ will eventually find a proof, otherwise $P_L$ will run forever. Thus $P_L$ corresponds to a Turing machine that halts on statements $X$ that are provable in $L$ and runs forever on other statements.

Thus the set of theorems in a logic $L$ is recursively enumerable.

A logic $L$ is *sound* if all theorems of $L$ are true. A logic is *effective* if the set of theorems is recursively enumerable. Let $S_L$ be $\{i : \text{in } L \text{ one can prove that Turing machine } T_i \text{ does not halt on input } i \}$. If the logic $L$ logic is effective then $S_L$ is recursively enumerable. It is reasonable to assume that logics are effective, using $P_L$ to partially decide the set of theorems.

**Theorem.** Suppose $L$ is a logic that is sound and effective. Then there is a Turing machine $T_j$ that does not halt on input $j$ but this fact cannot be proven in $L$.

**Proof.** Let $\Delta$ be $\{i : T_i \text{ does not halt on input } i \}$. We know that $\Delta$ is not recursively enumerable. Because $L$ is sound, $S_L \subseteq \Delta$. However, $S_L$ is recursively enumerable. Because $\Delta$ is not r.e., $S_L \neq \Delta$. Because $S_L \subseteq \Delta$, there is a $j$ such that $j \in \Delta$ but $j \notin S_L$. Thus there is a $j$ such that $T_j$ halts on input $j$ but this fact cannot be proven in $L$.

This shows that no finite logic can fully capture the non-halting of Turing machines. Even more, an integer $j$ as above can be constructed from $L$. Thus in any reasonable logic $L$ there is a statement (call it $X_L$) that is true but not provable in $L$.

This can also be presented using the *encode* notation as follows:

A logic $L$ is *sound* if all theorems of $L$ are true. A logic is *effective* if the set of theorems is recursively enumerable. Let $S_L$ be $\{\text{encode}(T) : \text{in } L \text{ one can prove that Turing machine } T \text{ does not halt on input } \text{encode}(T) \}$. If the logic $L$ logic is effective then $S_L$ is recursively enumerable. It is reasonable to assume that logics are effective, using $P_L$ to partially decide the set of theorems.

**Theorem.** Suppose $L$ is a logic that is sound and effective. Then there is a Turing machine $T$ that does not halt on input $\text{encode}(T)$ but this fact
cannot be proven in $L$.

**Proof.** Let $\Delta$ be \{\textit{encode}(T) : T does not halt on input $\text{encode}(T)$\}. We know that $\Delta$ is not recursively enumerable. Because $L$ is sound, $S_L \subseteq \Delta$. However, $S_L$ is recursively enumerable. Because $\Delta$ is not r.e., $S_L \neq \Delta$. Because $S_L \subseteq \Delta$, there is a $T$ such that $\text{encode}(T) \in \Delta$ but $\text{encode}(T) \not\in S_L$. Thus there is a $T$ such that $T$ halts on input $\text{encode}(T)$ but this fact cannot be proven in $L$.

This shows that no finite logic can fully capture the non-halting of Turing machines. Even more, a machine $T$ as above can be constructed from $L$. Thus in any reasonable logic $L$ there is a statement (call it $X_L$) that is true but not provable in $L$.

This leads to the Lucas paradox. Suppose $H$ is “human logic.” Suppose someday we learn what $H$ is. Then we can apply the above reasoning to construct $X_H$. We will then know that $X_H$ is true but not provable in $H$. But since we did this, we know that $X_H$ is true, so $X_H$ is provable in $H$.

Possibilities:
1. $H$ is unknowable by humans.
2. It is not possible to prove in $H$ that $X_H$ is true.
3. Humans can’t think straight. ($H$ is not sound.)
4. Our minds can perform operations that are not realizable on Turing machines.