How do we show that SATISFIABILITY $\leq_P$ 3SAT?

To show that SATISFIABILITY (SAT) is polynomial-time reducible to 3SAT, you need to demonstrate that you can transform any instance of SAT into an equivalent instance of 3SAT in polynomial time. In other words, you need to design a polynomial-time reduction from SAT to 3SAT.

Here's a high-level explanation of how this reduction can be performed:

1. **Input**: Start with an arbitrary instance of SAT, which consists of a Boolean formula in conjunctive normal form (CNF).

2. **Transformation**: To reduce this instance to 3SAT, you need to modify the CNF formula in such a way that it remains equivalent (i.e., has the same set of satisfying assignments) but is now in 3SAT format, which means each clause contains exactly three literals.

3. **Clause Transformation**: For each clause in the original CNF formula, you may need to introduce additional variables and clauses to ensure that each clause contains exactly three literals. This may involve converting some clauses into 3SAT format while keeping the overall formula equivalent.

4. **Example**: To illustrate, if you have a clause with more than three literals, say (A ∨ B ∨ C ∨ D), you can convert it into multiple 3SAT clauses. For instance:

- (A ∨ B ∨ X)
- (¬X ∨ C ∨ Y)
- (¬Y ∨ D ∨ E)
- (¬E) (to ensure the original clause is satisfied)

Here, X, Y, and E are new variables introduced to create these 3SAT clauses. This transformation ensures that the original clause and the new clauses are equivalent, and each clause contains exactly three literals.

5. **Completing the Reduction**: Repeat this process for each clause in the original CNF formula and for each variable in the original formula if necessary. The goal is to construct an equivalent 3SAT formula.

6. **Polynomial Time**: Ensure that this transformation can be performed in polynomial time. The number of new variables and clauses introduced is bounded by a polynomial in the size of the original SAT instance.

7. **Equivalence**: Prove that the 3SAT formula you obtained is equivalent to the original SAT formula, meaning they have the same set of satisfying assignments. This ensures that a solution to the 3SAT instance corresponds to a solution to the original SAT instance.

Once you've completed this transformation, you've shown a polynomial-time reduction from SAT to 3SAT. This demonstrates that 3SAT is at least as hard as SAT, and it's a fundamental result in computational complexity theory. It implies that if you can efficiently solve 3SAT, you can also efficiently solve SAT, making 3SAT NP-complete.