```
class Grammar:
    def __init__(self, filename):
        self.nonterminals = set()
        self.terminals = set()
        self.productions = {}
        self.start_symbol = None
        self.read_grammar(filename)

    def read_grammar(self, filename):
        with open(filename, 'r') as file:
            lines = file.readlines()

        for line in lines:
            line = line.strip()

            if not line:
                continue

            lhs, rhs = line.split('->')
            lhs = lhs.strip()
            rhs = [prod.strip() for prod in rhs.split('|')]

            # Add nonterminal (lhs)
            self.nonterminals.add(lhs)

            if self.start_symbol is None:
                self.start_symbol = lhs  # First nonterminal is the start symbol

            # Add productions for this nonterminal
            if lhs not in self.productions:
                self.productions[lhs] = []

            for prod in rhs:
                self.productions[lhs].append(prod.split())

                # Add symbols to terminals/nonterminals
                for symbol in prod.split():
                    if symbol.isupper():
                        self.nonterminals.add(symbol)
                    else:
                        self.terminals.add(symbol)

    def print_grammar(self):
        print("Nonterminals:", self.nonterminals)
        print("Terminals:", self.terminals)

        print("Productions:")

        for nonterminal, prods in self.productions.items():
            for prod in prods:
                print(f"{nonterminal} -> {' '.join(prod)}")

    def is_cfg(self):
        # Simple check for CFG (productions must have a single nonterminal on the left-hand side)
        for lhs in self.productions:
```

```
    if len(lhs) != 1 or not lhs.isupper():
        return False

return True
```