More MNT Examples

If I change this to \( n \geq 0 \), allowing \( 1 \) \( \epsilon \) \( L \), does the proof become wrong?

\[ L = \{ 0^n 1 0^n : n \geq 1 \} \text{?} \]

Some basic script:
Take \( S = 0^* \). Clearly \( S \) is infinite. Let any \( x, y \in S \) \( (x \neq y) \) be given. Then there are numbers \( m, n \geq 1 \) with \( m \neq n \).

\( S \) is infinite.

\[ \begin{align*}
& x = 1^n \text{ and } y = 1^n. \quad \text{Take } z = 10^m. \text{ Then:} \\
& xz = 0^n 1 0^n \in L \text{ but } yz = 0^n 1 0^m \notin L \text{ since } n \neq m. \quad \text{Hence } S \text{ is an infinite PD set for } L, \text{ so } L \text{ is not regular.} \checkmark
\end{align*} \]

(Study \( Q \): With \( L \), \( L^* \) above with \( n \geq 1 \), \( S = 0^* \) still PD?
\( \) The proof would be wrong, but can the proof be fixed?)

Footnote: \( \{ 0^n 0 1 0^n : n \geq 0 \} \) is an equivalent def'n of \( L \).

\[ L' = \{ 0^n 0^n : n \geq 1 \}\text{?} \]

Take \( S = 0^* \). Let any \( x, y \in S \) \( x \neq y \) be given. \( x = 0^m \text{ and } y = 0^n. \text{ Take } z = 0^n. \text{ Then:} \\
xz = 0^m 0^n \notin \; L' \ldots \text{no: it can still be } \epsilon \; L' \text{?} \\
L' = \{ 001^+ \} \text{ which is regular.} \]

\[ L'' = \{ \text{ww} \in W \; \text{w} \; \text{ww} : \text{ww} \; \text{30, 1?} \} \text{?} \]

"Critical Cases": \( W = 00000...01 \)

Take \( S = 0^* \). Clearly \( S \) is infinite!

Let any \( x, y \in S \) \( x \neq y \) be given. Then there are \( m, n \geq 1 \), \( m \neq n \), such that \( x = 0^m 1 \text{ and } y = 0^n 1 \). Take \( z = 0^m 1 \). Then:

\[ xz = 0^m 1 0^m 1 \in L'' \text{ by the division shown, but } yz = 0^n 1 0^n 1 \notin \]

because the only possible div is after the first 1, but \( n \neq m \) so it doesn't work.
II. The **class \( \text{RED} \)** (or **just \( \text{REG} \)) of **Regular Languages**:

\[
\begin{align*}
\text{string} &= \text{list}(<\text{char}>) \\
\text{language} &= \text{set}(<\text{string}>) \\
\text{First-order Objects} &= x, y, z, w, v, u, \ldots \\
\text{Second-order} &= L, A, B, C, D, \ldots \\
\text{Third-order} &= \text{set}<\text{language}> = \text{set}<\text{set}<\text{list}<\text{char}> >
\end{align*}
\]

The last few weeks have proved the following **Theorem**:

For any language \( L \subseteq \Sigma^* \), the following are equivalent:

(a) There is a regular expression \( R \) such that \( L = L(R) \)

(b) There is a DFA \( M \) such that \( L = L(M) \)

(c) There is an NFA \( N \) such that \( L = L(N) \)

(d) \( \cdots \textit{GNFA} \cdots \) etc!

**Proof:** (a) \( \rightarrow \) (c), (c) \( \rightarrow \) (b), (b, c, d) \( \rightarrow \) a.  

---

**Theorem 2:** For any regular expression \( R \), we can **build a regular expression \( R' \)** such that \( L(R') = \complement L(R) \).

**Abstract:** The class \( \text{RED} \) is **closed under complement**.

**Abstract proof:** Use (b) to take a DFA \( M \) st: \( L(M) = L \), then simply and quickly build \( M' = (\Sigma, \Sigma, S, s, \lambda, F) \).

**Concrete proof of Theorem 2:**

- (a) \( \rightarrow \) (b): Convert \( R \) into equivalent NFA \( N_R \).
- (c) \( \rightarrow \) (b): Convert \( N_R \) into equivalent DFA \( M_R \).

**Theorem 3:** \( \text{RED} \) is closed under \( \cap \).

- For any regular languages \( A, B \in \text{RED} \), the language \( A \cap B \) is also in \( \text{RED} \).

- **Proof:** We may take DFAs \( M_A, M_B \) st: \( L(M_A) = A \) and \( L(M_B) = B \). Then build a DFA \( M_C \) st: \( L(M_C) = L(M_A) \cap L(M_B) \) using "Cartesian Product" for DFAs.

Thus \( L(M_C) = A \cap B \) is a DFA, so \( A \cap B \) is regular.  

---

Let \( A, B \in \text{RED} \) be given.

**Proof:** We may take DFAs \( M_A, M_B \) st: \( L(M_A) = A \) and \( L(M_B) = B \). Then build a DFA \( M_C \) st: \( L(M_C) = L(M_A) \land L(M_B) \) using "Cartesian Product" for DFAs.
Suppose instead we are given regular expressions $\alpha$ and $\beta$, and we desire to build a regexp $\gamma$ st. $L(\gamma) = L(\alpha) \cap L(\beta)$?

Proof: 1. Convert $\alpha$, $\beta$ to equivalent NFAs (with 3 qres)
Algorithm: $N_\alpha$ and $N_\beta$ 
$L(N_\alpha) = L(\alpha)$, $L(N_\beta) = L(\beta)$.

2. Convert $N_\alpha$, $N_\beta$ to equivalent DFAs $M_\alpha$, $M_\beta$.
"Cart. Prod." $M_\gamma$ st. $L(M_\gamma) = L(M_\alpha) \cap L(M_\beta)$.

3. Convert $M_\gamma$ back to regexp $\gamma$ st. $L(\gamma) = L(M_\gamma)$.

**Puzzle Q:** Can we avoid expt. blowup of NFA to DFA? Not to mention $L$.

*: Since REG is closed under $\cap$ and $\cup$, it is closed under all Boolean operations.

**Theorem:** All finite languages are regular. Proof: If $L = \{w_1, w_2, \ldots, w_m\}$, then $L$ has the regexp $w_1 w_2 \ldots w_m$.

*: If $L$ is regular, then any language $L'$ obtained by adding or taking away finitely many strings is also regular.

**Because:** If $F$ is the finite language of strings whose status was changed, then $L' = (L \Delta F) \cup (F \cap L)$. [Example for all three objects: regular \cup \text{Bool op.} \text{ regular} \cup \text{F in text}(1)$\text{ or so}$]

For any $K \geq 1$, define $L_K = \{x \in \{0, 1\}^* : \text{bit } K \text{ from the end is a } '1'\}$.

**Regular Exp.** $R_K = (0+1)^* (0+1)^{K-1} : 12 + \lfloor \log_2 K \rfloor$ chars.

**NFA:** $N_K = \begin{array}{c}
01 \to \ldots \to \end{array}$

**DFA?** Study Fact: $\{0, 1\}^K$ is PD for $L_K$, so any DFA $M_K$ st. $L_K = L(M_K)$ needs $2^K$ states.