Reductions among String Problems

Richard Cole

November 7, 2007

Problem 1 \( A_{\text{Prog}} = \{ \langle P, w \rangle \mid P \text{ is a program, } w \text{ an input to } P \text{ and } P \text{ eventually halts on input } w \} \).

We have already shown that \( A_{\text{Prog}} \) is undecidable. We will revisit this proof.

Definition Let \( P \) be a program. Define \( L(P) = \{ x \mid x \text{ is an input to } P \text{ and } P \text{ halts on input } x \} \).

Problem 2 \( \text{REG} = \{ \langle P \rangle \mid P \text{ is a program and } L(P) \text{ is regular} \} \).

Claim \( A_{\text{Prog}} \) is reducible to \( \text{REG} \).

Proof Given a program \( P_{\text{REG}} \) to decide \( \text{REG} \), we construct a program \( P_{A_{\text{Prog}}} \) to decide \( A_{\text{Prog}} \).

Here is \( P_{A_{\text{Prog}}} \).

On input \( \langle P, w \rangle \):

Step 1: Construct a program \( Q \) such that \( L(Q) \) is regular if \( P \) halts on input \( w \), and \( L(Q) \) is not regular otherwise. In particular, if \( L(Q) \) is regular it will be \( (a \cup b)^* \), and otherwise it will be the non-regular language \( A = \{ a^i b^i \mid i \geq 0 \} \). \( Q \) is the following program:

On input \( x \):

- if \( x = a^i b^i \), for some \( i \geq 0 \), then halt;
- otherwise run \( P \) on input \( w \).

We see that if \( P \) halts on input \( w \), \( Q \) halts on all its inputs, namely on \( (a \cup b)^* \). While if \( P \) does not halt on input \( w \), \( Q \) halts on exactly \( A \).

Step 2 Run \( P_{\text{REG}}(\langle Q \rangle) \) and output \( P_{\text{REG}} \)'s output.

\( P_{\text{REG}} \) outputs accept if \( L(Q) \) is regular, which is the case if \( P \) halts on input \( w \), while \( P_{\text{REG}} \) outputs reject if \( P \) does not halt on input \( w \).

\( \square \)

Corollary \( \text{REG} \) is not decidable.

Problem 3 \( E_{\text{Prog}} = \{ \langle P \rangle \mid P \text{ does not halt on any input} \} \).

Claim \( A_{\text{Prog}} \) is reducible to \( E_{\text{Prog}} \).

Proof We need to show how, given a program \( P_{E_{\text{Prog}}} \) to decide \( E_{\text{Prog}} \), we can construct a program \( P_{A_{\text{Prog}}} \) to decide \( A_{\text{Prog}} \). Here is \( P_{A_{\text{Prog}}} \).
On input \((P, w)\):

**Step 1.** Construct a new program \(Q_w\) which behaves as follows.

Definition of \(Q_w\):
- On input \(x\): if \(x \neq w\) then loop forever. Otherwise run \(P\) on input \(w\).

**Step 2** Run \(P_{\text{Eprog}}(\langle Q_w \rangle)\), and output its result reversed (output accept if \(P_{\text{Eprog}}\) rejects, output reject if \(P_{\text{Eprog}}\) accepts).

Note that \(Q_w\) halts on at most one input, namely \(w\). It halts on input \(w\) exactly if \(P\) halts on input \(w\). Thus \(Q_w \notin E_{\text{Prog}}\) if \(P\) halts on input \(w\), and \(Q_w \in E_{\text{Prog}}\) if \(P\) does not halt on input \(w\). Consequently \(P_{A_{\text{Prog}}}\) decides \(A_{\text{Prog}}\).

\(\square\)

**Corollary** \(E_{\text{Prog}}\) is not decidable.

**Problem 4** \(EQ_P = \{ \langle P_1, P_2 \rangle \mid L(P_1) = L(P_2) \}\).

**Claim** \(EQ_P\) is reducible to \(E_{\text{Prog}}\).

**Proof** Given a program \(P_{EQ}\) to decide \(EQ_P\) we need to show how to construct a program \(P_{E_{\text{Prog}}}\) to decide \(E_{\text{Prog}}\). Here is \(P_{E_{\text{Prog}}}\):

- **Step 1** Construct program \(N\) which loops forever on every input.

- **Step 2** Run \(P_{EQ}(\langle P, N \rangle)\) and output its answer.

  Note that \(L(N) = \phi\). \(P_{EQ}\) accepts exactly if \(L(P) = L(N)\), i.e. if \(L(P) = \phi\). But \(L(P) = \phi\) exactly if \(\langle P \rangle \in E_{\text{Prog}}\). Thus \(P_{E_{\text{Prog}}}\) accepts \(E_{\text{Prog}}\).

\(\square\)

**Corollary** \(EQ_P\) is not decidable.