Exercise 64.

Claim: The improved semi-decision procedure is sound and complete.

Proof:

1. Soundness: If there is n, such that $s_{n-1} \rightarrow^*_{R_n} s_n$ and $t_{n-1} \rightarrow^*_{R_n} t_n$, and $s_n = t_n$, then $s \approx_E t$. In order to prove this, use induction on n and prove Hint 1.

2. Completeness: If $s \approx_E t$, then there is n, such that $s_n = t_n$.

   In order to prove this, notice that since R_∞ is terminating and confluent, s has a normal form $\hat{s}$ in R_∞, and t has a normal form $\hat{t}$ in R_∞ and $\hat{s} = \hat{t}$. Then if we can prove Hint 2, s_n must be equal to $\hat{s}$ and t_n must be equal to $\hat{t}$. Otherwise, $s_n \rightarrow^*_{R_\infty} \hat{s}$ (because R_∞ is confluent), and there is k, such that $s_n \rightarrow^*_{R_k} \hat{s}$, which is contrary to Hint 2. The same about t_n (if it is not reducible by any R_m, where $m \geq n$, then it should not be reducible by any R_k).

   So, you have to prove Hint 2, in order to complete argument. It is an easy proof by contradiction: if there is no such s_n or t_n, we can construct an infinite chain of reductions from s and t, contradicting the fact that R_∞ is terminating.

Exercise 65. (b)

The polynomial order in the exercise requires $A = \mathbb{N}\backslash\{0,1\}$
We get $R_0 = \{\ x + (y + z) \rightarrow (x + y) + z,\ \ .$
$f(x) + f(y) \rightarrow f(x + y)\ \ \}$
We have a critical pair obtained from $1 \leftarrow 1$, at position $p = 2$, with substitution $[y \mapsto x', z \mapsto (y' + z')]$, but it is joinable. We have another one: $1 \leftarrow 2, p = 2, [y \mapsto f(x'), z \mapsto f(y')]$ which results in adding one rule: $(x + f(x')) + f(y') \rightarrow x + f(x' + y')$.

Exercise 66.

The procedure can be written as follows:
$R := Orient\&Delete(G_0)$
While L-Simplify-Rule is applicable to R, do
let $l \rightarrow v \in R$, $l \overset{\sqsupset}{\rightarrow}_R u$.
Since $>$ is assumed to be total on ground terms:
$v > u$ or $v = u$ or $u > v$.

- if $u = v$ then $R := R\backslash\{l \rightarrow v\}$
- if $u > v$ then $R := R\backslash\{l \rightarrow v\} \cup \{u \rightarrow v\}$
- if $v > u$, then $R := R\backslash\{l \rightarrow v\} \cup \{v \rightarrow u\}$

od. (end of *while* loop)

Return R.

(a) The procedure always terminates. To show this, let treat rules as pairs (l, r) compared lexicographically with the assumed reduction order, total on ground terms.

Let treat R as a mutliset of rules.

Claim: if R' is obtained after 1 run of *while* loop, then $R >_{mul} R'$.

In the proof of the claim, we have to consider all cases $v > u$ or $v = u$ or $u > v$ and show that $R >_{mul} R'$.

For example, if $v > u$, then $R' = R - (l, v) \cup (u, v)$. But $l > v$, so $R >_{mul} R'$.

The termination follows from well-foundedness of $>_{mul}$.

(b) The procedure is fair (i.e. all critical pairs are computed).

To show this assume that $l_1 \to r_1, l_2 \to r_2$ are in R, and there is a critical overlap between these two rules. Since we have only ground terms, $l_1|_p = l_2$. The critical pair, we want to compute is: $< r_1, l_1[r_2]_p >$.

But in this case L-Simplify-Rule applies to $l_1 \to r_1$, because $l_1 \stackrel{\exists}{\to} l_1[r_2]_p$ by rule $l_2 \to r_2$. Hence our procedure will replace $l_1 \to r_1$, by $r_1 \to l_1[r_2]_p$, if $r_1 > l_1[r_2]_p$, or $l_1[r_2]_p \to r_1$, if $l_1[r_2]_p > r_1$ and it will remove the rule if $r_1 = l_1[r_2]_p$.

(c) and (d) The procedure is correct and doesn't fail.

It doesn't fail, because $>$ is total and hence E is always empty. Since it terminates always, and obviously it terminates with R convergent and equivalent to E, all its runs are successful, and hence the procedure is correct.