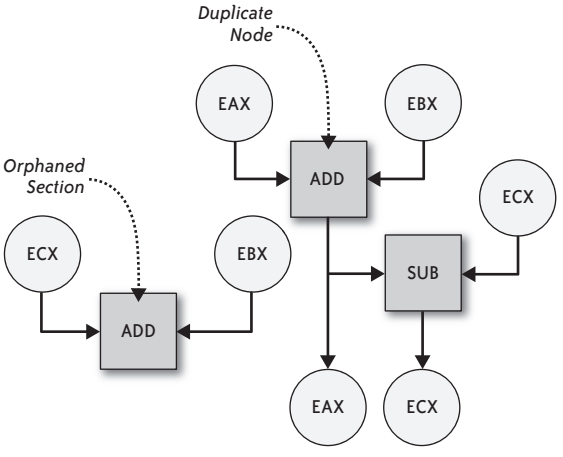

FIGURE 9-10. Directed acyclic graph features

TIP #2: STATIC IS GOOD

If a method can be made static, then make it so. Static methods are not virtual, and so are not dispatched dynamically. Advanced VMs can inline such methods much more easily and readily than instance methods.

Handling exceptions

Now that we can handle the compilation of basic blocks, we must restore some of the ugliness to the situation. As we discussed previously, exceptions are not always errors. Page faults and protection violations are thrown with abandon as a matter of course. When an exception is thrown, the processor state must be consistent up to the last successful execution of an operation. This obviously has rather severe implications for the compiler. Having mapped exceptions in IA-32 to Java exceptions, we know that the only practical solution to this problem is to catch the exception and, once inside the exception handler, ensure that the state is consistent with the last successful operation.

The behavior of an exception path within any given basic block is just like that of the basic block itself. It has one entry point and one exit point, and the only difference from the main path is the exit point. As the exception path is not so very different from the basic block itself, the most natural way to represent it is with its own directed acyclic graph. The graph for the exception path will share the same node set as the basic block, but will have a distinct set of sinks mapped to its own distinct exit point.