Halt-no-Input

Prove that the Halt-no-Input problem is undecidable.

Halt-no-Input problem: given a program `P` that requires no input, does `P` halt?

We want to prove that the Halt-no-Input problem is undecidable. Therefore, we assume the existence of a function that solves this problem. Suppose `HaltNI` is a function that computes a solution to the Halt-no-Input problem:

```
let HaltNI (P: unit -> 'a) : bool = ...
```

Although the problem does not state precisely what type `P` is, we are allowed to make reasonable assumptions. For example, programs are in some sense functions, which is why `P` has the type `unit -> 'a` above.

Let's now construct a reduction.

```
let Halt (P: string -> 'a)(i: string) : bool =
  let P'() = P i
  HaltNI P'
```

The above reduction reduces Halt-no-Input to the Halting Problem. It works by showing that we can solve the Halting Problem by changing the form of the problem slightly, then by calling `HaltNI`. However, we know that we cannot solve the Halting Problem, because it is undecidable. Since all of the above code is perfectly legal F#, then the only flaw in our logic must have been our assumption that `HaltNI` is decidable. Therefore, `HaltNI` is undecidable.

Alternatively, we can think of programs as `string`s if it is more convenient, and we will use string manipulation to construct our reduction. Therefore, we make a different set of reasonable assumptions about `HaltNI`:

```
let HaltNI (P: string) : bool = ...
```

Suppose `HaltNI` expects that `P` be a function of the form `let MyFunc () = ...` Then we can construct a different reduction:

```
let Halt (P: string)(i: string) : bool =
  let P' = "let MyFunc2 () = \n\t" + P + "\n\tMyFunc \"" + i + "\""
  HaltNI P'
```

where `Halt` takes a `P` of the form `let MyFunc (i:  string) = ...` For example, if we call `Halt` with `P` defined as `let MyFunc (i:  string) = "hello " + i` and `i` defined as `"world"`, then the reducer constructs the string `P'`,

```
  let MyFunc2 () =
    let MyFunc (i: string) = "hello " + i
    MyFunc "world"
```

in order to convert the problem into the form expected by `HaltNI`. Since string manipulation is clearly legal in F#, we must again conclude that `HaltNI` is undecidable.

Observe that in both proofs, the reduction does not solve the Halt-no-Input problem directly. Instead, the reduction algorithm changes the form of the problem, delegating computation of the solution to the function that we assumed exists (`HaltNI`).