signature, the result of this process is simply *A*. Otherwise, type arguments for *A* are inferred from *B* producing an instantiation of *A* that can be related to *B*:

- Using the process described in 3.8.6, inferences for *A*'s type parameters are made from each parameter type in *B* to the corresponding parameter type in *A* for those parameter positions that are present in both signatures, where rest parameters correspond to an unbounded expansion of optional parameters of the rest parameter element type.
- The inferred type argument for each type parameter is the best common type (section 3.10) of the set of inferences made for that type parameter.
- Provided all inferred type arguments satisfy their corresponding type parameter constraints, the result is an instantiation of *A* with the inferred type arguments.
- Otherwise, *A* cannot be instantiated in the context of *B* and the process is unsuccessful.

Given the declarations

```
var f: (x: string) => string[];
var g: <T>(x: T) => T[];
```

the assignment 'f = g' causes the signature of 'g' to be instantiated in the context of the signature of 'f', which causes inferences for 'T' to be made from the parameter type 'string' to the parameter type 'T', ultimately producing an assignment compatible instantiation of 'g' with the type argument 'string'.

Changing the example to

```
var f: (x: { a: string; b: number }) => {}[];
var g: <T>(x: { a: T; b: T }) => T[];
```

inferences for 'T' are now made from the parameter type '{ a: string; b: number }' to the parameter type { a: T; b: T }', leading to inferences of 'string' and 'number' for 'T'. The best common type of those two types is the empty object type, which becomes the type argument for the resulting assignment compatible instantiation of 'g'.

In the example

```
var f: <T>(x: T, y: T) => { x: T; y: T };
var g: <U, V>(x: U, y: V) => { x: U; y: V };
```

the assignment 'f = g' causes 'T' to be inferred for both 'U' and 'V', thus producing an assignment compatible instantiation of 'g'. The reverse assignment 'g = f' leads to inferences of 'U' and 'V' for 'T', producing an instantiation of 'f' with the empty object type as an argument, and since an empty object type is not assignable to a type parameter, the assignment is an error.

3.8.6 Type Inference

In certain contexts, inferences for a given set of type parameters are made *from* a type *S*, in which those type parameters do not occur, *to* another type *T*, in which those type parameters do occur. Inferences