1 Example program

Listing 1 gives an example program. Line 1 declares a new variable `x`, initialized to `42`. Lines 3-5 declare a function `mut`, which closes over `x`. Its body sets `x` to the string `"hello"`. Line 7 represents arbitrary additional program code. Line 9 finally invokes `mut()`, with the effect of assigning `x = "hello"`.

Listing 1 Program with a failing type cast

```
1  var x = 42;
2
3  function mut() {
4      x = "hello";
5  }
6
7  // ...other program code here
8
9  mut();
10
11 (x: string);  // ERROR
```

In Line 11, the programmer attempts to cast `x` to type `string`. This is equivalent to asserting that the type of `x` is a subtype of `string`.

This type cast gives the following error:

```
9: (x: string);
    ^ Cannot cast `x` to string because number [1] is incompatible with string [2].
References:
1: var x = 42;
           ^ [1]
9: (x: string);
```

This error states that `x` cannot be cast to `string`, because it may be a `number`. In the following section, we step through the derivation at a high level.

2 Derivation at a glance

Table 1: Simplified derivation for the program in Listing 1

Code	Γ	Constraints
`let x = 42;`	$\boxed{x : (\text{number}, \tau_0)}$	$\boxed{\text{number} <: \tau_0}$

`function mut() {`
`  x = "hello";`
`}`
	$x : (\text{number}, \tau_0)$
	$\boxed{mut : (\text{void} \xrightarrow{x} \text{void}, \sigma_0)}$
		$\text{number} <: \tau_0$
		$\boxed{\text{string} <: \tau_0}$
		$\boxed{\text{void} \xrightarrow{x} \text{void} <: \sigma_0}$

Code	Γ	Constraints
`mut()`	$x : (\boxed{\alpha}, \tau_0)$ $mut : ...$	$\text{number} <: \tau_0, \quad \boxed{\alpha <: \tau_0}$ $\text{string} <: \tau_0$ $\text{void} \xrightarrow{x} ...$
	$x : (\alpha, \tau_0)$ $mut : ...$	$\text{number} <: \tau_0, \quad \alpha <: \tau_0, \quad \boxed{\tau_0 <: \alpha}$ $\text{string} <: \tau_0$ $\text{void} \xrightarrow{x} ...$

Casting `(x: string)` in the final line adds a constraint of the form

$$\frac{\Gamma \vdash x : (\alpha, \tau_0)}{\alpha <: \text{string}}$$

saying that whatever type α is currently assigned to x in Γ, must be a subtype of `string`.

Following the derivation in the above table, our environment at this point in the program will prove that

$$\Gamma(x) = (\alpha, \tau_0) \qquad\qquad \text{number} <: \tau_0 = \alpha$$

However, adding in $\alpha <:$ string triggers a contradiction:

$$\begin{aligned} \text{number} <: \tau_0 &= \alpha \\ &<: \text{string} \end{aligned}$$

By transitivity, this implies that `number <: string`, which is clearly impossible.

Another way to interpret this error is to read the constraints

$$\begin{aligned} \text{number} <: \tau_0 &= \alpha \\ \text{string} <: \tau_0 &= \alpha \end{aligned}$$

as a statement that α is a union of `number | string`. As such, we cannot possibly assert that $\alpha = \text{string}$, since α might well be `number`.