The macro `extrapolate` computes a "superpath" (as opposed to "subpath") for a single Bézier segment in such a way that the following identity holds (for $0 \leq t_1 \leq t_2 \leq 1$):

$$\text{subpath}(t_1, t_2) \text{ of } \text{extrapolate}(t_1, t_2) \text{ of } b = b$$

Below, there are results of the command `extrapolate(.3, .7) of p` for three similarly defined paths. The black line denotes the source path, the gray one—its extrapolation.

$$p = (0, 0) \{\text{right}\} \ldots \{\text{up}\}(s, s);$$

Exercise 1. What happens if the relation $0 \leq t_1 \leq t_2 \leq 1$ is not fulfilled? (Hint: there are a few possible cases.)

Exercise 2. True or false:

$$\text{point 1 of } \text{extrapolate}(t_a, t) \text{ of } b = \text{point 1 of } \text{extrapolate}(t, t_b)$$

for $t_a <> t_b$

Exercise 3. Try to imagine the result of the extrapolation for such weird (yet trivial) paths as:

$$(0, 0) \ldots \text{controls}(0, 0) \text{ and } (100, 0) \ldots (100, 0)$$

or

$$(0, 0) \ldots \text{controls}(100, 0) \text{ and } (0, 0) \ldots (100, 0)$$

```
var def extrapolate expr t of b = \% t pair, b Bézier segment
clearxy;
Casteljau(xpart(t)) = point 0 of b;
Casteljau(1/3 [xpart(t), ypart(t)]) = point 1/3 of b;
Casteljau(2/3 [xpart(t), ypart(t)]) = point 2/3 of b;
Casteljau(ypart(t)) = point 1 of b;
z_0 \ldots \text{controls} z_1 \text{ and } z_2 \ldots z_3
enddef;
\%
def Casteljau(expr t) =
t[t[t[z_0, z_1], t[z_1, z_2]], t[t[z_1, z_2], t[z_2, z_3]]]
enddef;
```