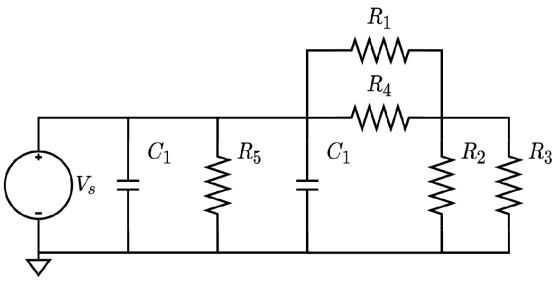
Simplifying circuits

ELAB can recursively simplify your circuit, updating the entire circuit object in the process.

We start by loading a circuit from a text file and displaying its netlist. This circuit is arbitrary and could be of any size and shape. The circuit is then analyzed and some results are displayed.

```
circuit = Circuit('circuits/series_parallel.txt');
circuit.list
```

```
ans =
    'V1 1 0 AC 10
     R1 1 2 1000
     R2 2 0 2000
     R3 2 0 2000
     R4 1 2 3000
     R5 1 0 1000
     C1 1 0 2
     C2 0 1 3
     '
```




```
ELAB.analyze(circuit)
```

```
Symbolic analysis successful (0.427216 sec).
```

```
circuit.symbolic_node_voltages
```

```
ans =
```

$$\begin{pmatrix} v_1 = V_1 \\ v_2 = \dfrac{R_2\,R_3\,V_1\;(R_1+R_4)}{R_1\,R_2\,R_3+R_1\,R_2\,R_4+R_1\,R_3\,R_4+R_2\,R_3\,R_4} \end{pmatrix}$$

We then simplify the circuit and repeat the process. Notice, that the naming and orientation of individual elements are accounted for.

```
circuit.Resistors(1).resistance
```

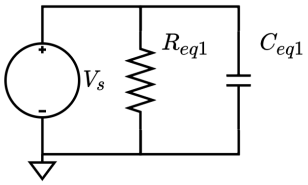
```
ans = 1000
```

```
circuit.Resistors(2).resistance
```

```
ans = 2000
```

```
ELAB.simplify(circuit);
circuit.list
```

```
ans =
    'V1 1 0 AC 10
     Req1 1 0 7000/11
     Ceq1 1 0 5
     '
```



ELAB recursively simplified the series and parallel resistors and capacitors, calculating their new values and giving them new names. Since all the resistors and capacitors can be reduced to a single resistor and capacitor in parallel, the node voltage at node 1 is simply the source voltage.

```
ELAB.analyze(circuit)
```

```
Symbolic analysis successful (0.129734 sec).
```

```
circuit.symbolic_node_voltages
```

ans = $v_1 = V_1$

Let's check if it was done right. From the pre-simplify diagram, we see that the resistors simplify to:

$$
\begin{aligned}
R_{\text{eq1}} &= ((R_1||R_4) + (R_2||R_3))||R_5 \\
&= \frac{\left(\left(\frac{1k\Omega \cdot 3k\Omega}{1k\Omega + 3k\Omega}\right) + \left(\frac{2k\Omega \cdot 2k\Omega}{2k\Omega + 2k\Omega}\right)\right) \cdot 1k\Omega}{\left(\left(\frac{1k\Omega \cdot 3k\Omega}{1k\Omega + 3k\Omega}\right) + \left(\frac{2k\Omega \cdot 2k\Omega}{2k\Omega + 2k\Omega}\right)\right) + 1k\Omega} \\
&= 7k\Omega/11
\end{aligned}
$$

And the capacitors simplify to:

$$
\begin{aligned}
C_{\text{eq1}} &= C_1||C_2 \\
&= C_1 + C_2 \\
&= 5
\end{aligned}
$$

ELABorate can simplify series and parallels of any 2-terminal element.

Here are examples of how a larger number of series and parallels are handled.

```
circuit = Circuit('circuits/triple_parallel.txt');
circuit.list
```

```
ans =
    'Vs 1 0 DC 12
     R1 1 0 3000
     R2 1 0 4000
     R3 1 0 2000
     '
```

```
ELAB.simplify(circuit);
circuit.list
```

```
ans =
    'Vs 1 0 DC 12
     Req1 1 0 12000/13
     '
```

```
circuit = Circuit('circuits/triple_series.txt');
circuit.list
```

```
ans =
    'Vs 1 0 DC 12
     R1 1 2 3000
     R2 2 3 4000
     R3 3 0 2000
     '
```

```
ELAB.simplify(circuit);
circuit.list
```

```
ans =
    'Vs 1 0 DC 12
     Req1 1 0 9000
     '
```

```
circuit = Circuit('circuits/quad_series.txt');
circuit.list
```

```
ans =
    'Vs 1 0 DC 12
     R1 1 2 3000
     R2 2 3 4000
     R3 3 4 2000
     R4 4 0 1000
     '
```

```
ELAB.simplify(circuit);
circuit.list
```

```
ans =
    'Vs 1 0 DC 12
     Req1 1 0 10000
     '
```

The simplify function also ensures approriate naming of the new equivalent elements, even if the elements cannot be simplified down to one element.

```
circuit = Circuit('circuits/multiple_eqs.txt');
circuit.list
```

```
ans =
    'Vs 1 0 DC Vs
     R1 1 2 R1
     R2 2 3 R2
     R3 4 5 R3
     R4 4 5 R4
     R5 5 0 R5
```

```
     C1 3 4 C1
     '
```

```
ELAB.simplify(circuit);
circuit.list
```

```
ans =
    'Vs 1 0 DC Vs
     Req1 1 2 R1 + R2
     Req2 3 0 R5 + (R3*R4)/(R3 + R4)
     C1 2 3 C1
     '
```