1. Objective:

We are to design an expert system to simplify electrical circuits by applying reduction rules and derive equivalent resistance, R_{in}.

2. Knowledge Acquisition:

a) The knowledge needed is a situation-action based knowledge.

- Situation: A circuit of resistors with their respective values were provided.
- Action: Implement appropriate network reduction rule for the specified electrical circuit.

b) Steps in modelling of the system

- **Identification:** The first step in acquiring knowledge for an expert system is to characterize the important aspect of the problem. For this circuit, we are to apply reduction rules to a given circuit of resistors to obtain the equivalent resistance.
- **Conceptualization:** The circuit structure is defined using a frame-based system which allowed for incorporation of structural knowledge into the knowledge base. Rules are defined to search through the knowledge base to reduce the specified networks.
- **Formalization:** This process involves mapping the key concepts, sub problems, and information flow characteristics isolated during conceptualization into more formal representations based on various knowledge engineering tools or frameworks. The template for the network is created and reduction rules were formulated.
- **Implementation:** Implementation involves mapping the formalized knowledge from the previous stage into the representational framework associated with the tool chosen for the problem. The prototype knowledge base is implemented by using JESS—a rule-based inference engine.
- **Testing:** The testing stage involves evaluating the prototype system and the representation forms used to implement it. Another circuit was developed as directed graph to test the expert system.

3. Modelling:

a) The data structure of the resistors is modelled as follows:

Name, node1, node2, value

```
(Resistor R1) (Node_1 A) (Node_2 B) (Res_value 2)
```

b) The electric network provided was modelled as a directed graph. The direction of the arc expresses the order of the arguments (nodes) to the predicate symbol without redefining the same node twice. This enables us to define the rules without priority. The orientation of the arc can be reversed if necessary, to corresponds to the direction of flow in the network.

For the first circuit:

```
;;Assign values to each resistor
(deffacts circuit1
    (R (Resistor R1) (Node_1 A) (Node_2 B) (Res_value 2))
    (R (Resistor R2) (Node_1 B) (Node_2 C) (Res_value 2))
    (R (Resistor R3) (Node_1 B) (Node_2 D) (Res_value 4))
    (R (Resistor R4) (Node_1 C) (Node_2 D) (Res_value 3))
    (R (Resistor R5) (Node_1 C) (Node_2 E) (Res_value 4))
    (R (Resistor R6) (Node_1 D) (Node_2 E) (Res_value 2)))
```

For the second circuit:

```
(deffacts circuit2
(R (Resistor R1) (Node_1 A) (Node_2 B) (Res_value 2))
(R (Resistor R2) (Node_1 B) (Node_2 E) (Res_value 5))
(R (Resistor R3) (Node_1 B) (Node_2 D) (Res_value 1))
(R (Resistor R4) (Node_1 B) (Node_2 C) (Res_value 4))
(R (Resistor R5) (Node_1 C) (Node_2 D) (Res_value 4))
(R (Resistor R6) (Node_1 D) (Node_2 E) (Res_value 3))
(R (Resistor R7) (Node_1 A) (Node_2 C) (Res_value 2))
(R (Resistor R8) (Node_1 I) (Node_2 C) (Res_value 4))
(R (Resistor R9) (Node_1 L) (Node_2 H) (Res_value 3))
(R (Resistor R10) (Node_1 H) (Node_2 I) (Res_value 1))
(R (Resistor R11) (Node_1 J) (Node_2 H) (Res_value 5))
(R (Resistor R12) (Node_1 A) (Node_2 J) (Res_value 4))
(R (Resistor R13) (Node_1 J) (Node_2 I) (Res_value 1))
(R (Resistor R14) (Node_1 A) (Node_2 I) (Res_value 2))
(R (Resistor R15) (Node_1 A) (Node_2 K) (Res_value 4))
(R (Resistor R16) (Node_1 A) (Node_2 L) (Res_value 3))
(R (Resistor R17) (Node_1 A) (Node_2 K) (Res_value 1))
(R (Resistor R18) (Node_1 K) (Node_2 M) (Res_value 2))
(R (Resistor R19) (Node_1 L) (Node_2 I) (Res_value 4))
(R (Resistor R20) (Node_1 M) (Node_2 I) (Res_value 5)))
```

c) The rules for the reduction rules were defined as follow:

- Series: For two resistors connected in series as shown below, the equivalent resistance is given by:

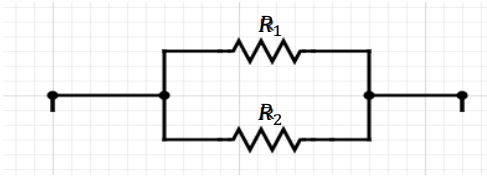
$$R_{eq} = R_1 + R_2$$

Logic used:

```
;DEFINE RULES
;;Series Reduction
(defrule series
     ?R1 <- (R
          (Resistor ?n1)
          (Node_1 ?Node_1)
          (Node_2 ?Node_2)
          (Res_value ?r1))
     ?R2 <- (R
          (Resistor ?n2)
          (Node_1 ?Node_2)
          (Node_2 ?node3)
          (Res_value ?r2))
     (not
          (or
               (R
                    (Resistor ?n4&~?n2&~?n1)
                    (Node_1 ?Node_2))
               (R
                    (Resistor ?n4&~?n1&~?n2)
                    (Node_2 ?Node_2))))
     =>
     (modify ?R1
          (Node_2 ?node3)
          (Res_value (+ ?r1 ?r2)))
     (retract ?R2))
```

- Parallel: For two resistors connected in parallel as shown below, the equivalent resistance is given by

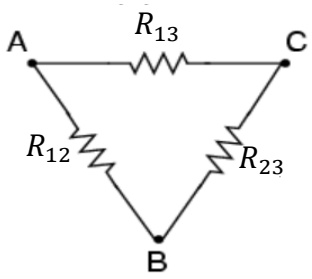
$$R_{eq} = \frac{R_1 R_2}{R_1 + R_2}$$

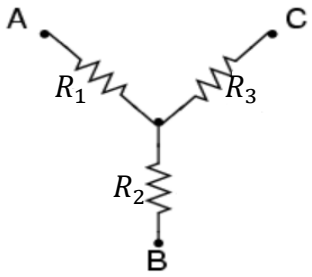


Logic used:

```
;; Parallel Reduction
(defrule parallel
     ?R1 <- (R(Resistor ?n1)
          (Node_1 ?Node_1)
          (Node_2 ?Node_2)
          (Res_value ?r1))
     ?R2 <- (R(Resistor ?n2)
          (Node_1 ?Node_1)
          (Node_2 ?Node_2)
          (Res_value ?r2))
     (test (neq ?n1 ?n2))
     =>
     (modify ?R1
(Res_value
               (/ (* ?r1 ?r2)(+ ?r1 ?r2))))
     (retract ?R2))
```

- Delta-star conversion: For resistors connected in delta (Δ), they are converted to star (Y) using the following formulation.





$$R_1 = \frac{R_{12}R_{13}}{R_{12}+R_{13}+R_{23}};\ R_2 = \frac{R_{12}R_{23}}{R_{12}+R_{13}+R_{23}};\ Rc = \frac{R_{13}R_{23}}{R_{12}+R_{13}+R_{23}}$$

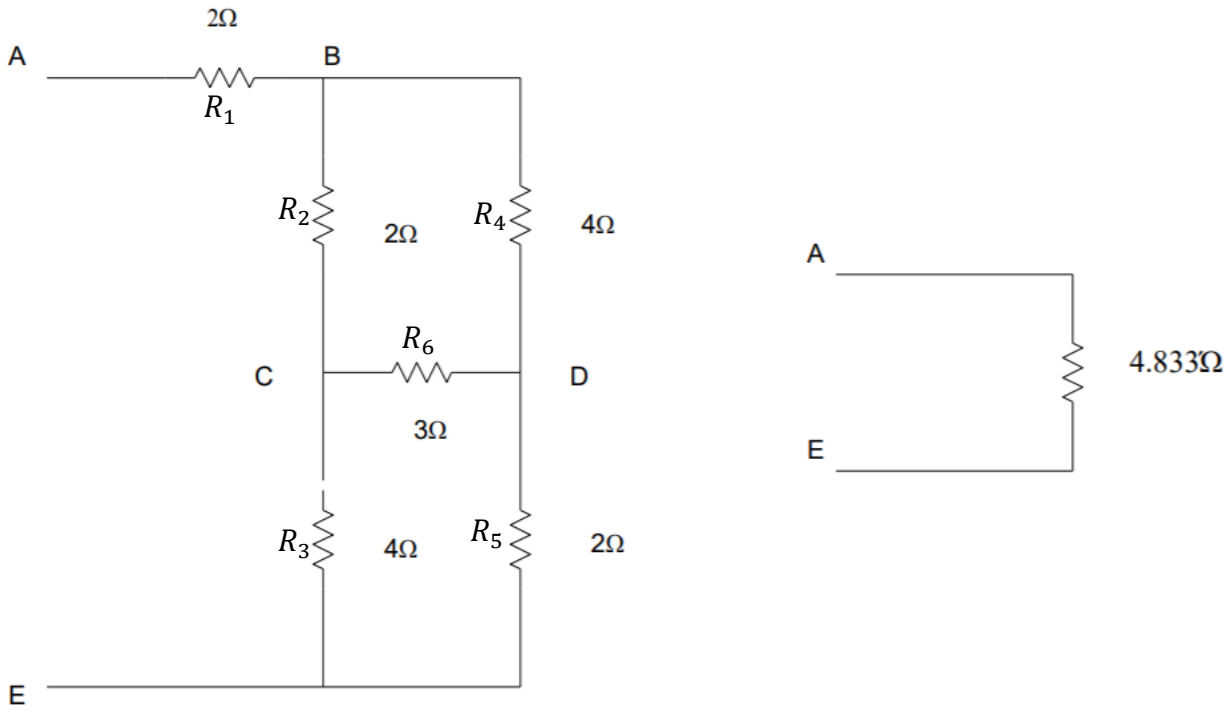
Logic used:

```
;; Delta-Star Transformation
(defrule delta-star
     ?R1 <- (R(Resistor ?n1)
          (Node_1 ?Node_1)
          (Node_2 ?Node_2)
          (Res_value ?r1))
     ?R2 <- (R
          (Resistor ?n2)
          (Node_1 ?Node_2)
          (Node_2 ?node3)
          (Res_value ?r2))
     ?R3 <- (R
          (Resistor ?n3)
          (Node_1 ?Node_1)
          (Node_2 ?node3)
          (Res_value ?r3))
     (or
          (R
               (Resistor ?n4 & ~?n1&~?n2&~?n3)
               (Node_2 ?Node_2)))
     =>
     (modify ?R1
     (Node_2 (str-cat ?Node_1 ?Node_2))
          (Res_value (/ (* ?r1 ?r3) (+ ?r1 ?r2 ?r3))))
     (modify ?R2
          (Node_2 (str-cat ?Node_1 ?Node_2))
          (Node_1 ?Node_2)
          (Res_value (/ (* ?r1 ?r2) (+ ?r1 ?r2 ?r3))))
     (modify ?R3
          (Node_1 (str-cat ?Node_1 ?Node_2))
          (Res_value (/ (* ?r2 ?r3) (+ ?r1 ?r2 ?r3)))))
```

4. Implementation:

The expert system that simplifies the circuits was implemented in JESS (available in the appendix)

For the first circuit:



Result:

```
;;Assign values to each resistor
(deffacts data
     (R (Resistor R1) (Node_1 A) (Node_2 B) (Res_value 2))
     (R (Resistor R2) (Node_1 B) (Node_2 C) (Res_value 2))
     (R (Resistor R3) (Node_1 C) (Node_2 E) (Res_value 4))
     (R (Resistor R4) (Node_1 B) (Node_2 D) (Res_value 4))
     (R (Resistor R5) (Node_1 D) (Node_2 E) (Res_value 2))
     (R (Resistor R6) (Node_1 C) (Node_2 D) (Res_value 3)))

;DEFINE RULES
;;Series Reduction
(defrule series
     ?R1 <- (R
```

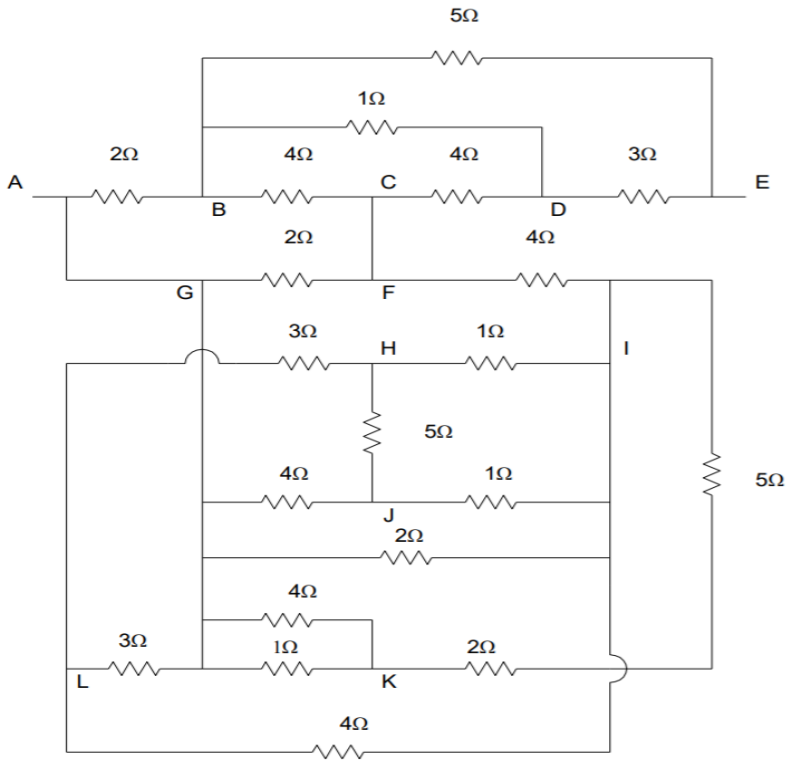
Problems | Javadoc | Declaration | Console

```
<terminated> Circuit.clp [Jess Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (2020-03-07 11:47:

Jess, the Rule Engine for the Java Platform
Copyright (C) 2008 Sandia Corporation
Jess Version 7.1p2 11/5/2008

The equivalent resistance, Req, for the circuit = 4.8333333333333330hms
```

For the second circuit:



Result:

```
(deffacts circuit2
(R (Resistor R1) (Node_1 A) (Node_2 B) (Res_value 2))
(R (Resistor R2) (Node_1 B) (Node_2 E) (Res_value 5))
(R (Resistor R3) (Node_1 B) (Node_2 D) (Res_value 1))
(R (Resistor R4) (Node_1 B) (Node_2 C) (Res_value 4))
(R (Resistor R5) (Node_1 C) (Node_2 D) (Res_value 4))
(R (Resistor R6) (Node_1 D) (Node_2 E) (Res_value 3))
(R (Resistor R7) (Node_1 A) (Node_2 C) (Res_value 2))
(R (Resistor R8) (Node_1 I) (Node_2 C) (Res_value 4))
(R (Resistor R9) (Node_1 L) (Node_2 H) (Res_value 3))
```

Problems | Javadoc | Declaration | Console

```
<terminated> Circuit.clp [Jess Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (2020-03-11 12:29:21

Jess, the Rule Engine for the Java Platform
Copyright (C) 2008 Sandia Corporation
Jess Version 7.1p2 11/5/2008

The equivalent resistance, Req, for the circuit = 3.28769291810878660hms
```

5. Questions:

a) The circuit needs to be modelled as a directed graph to ensure that network is reduced from branch to branch to prevent counterproductive revisiting of nodes which have been effectively analyzed. Alternatives to a directed graph includes the usage of directed acyclic graph or the usage of trees.

b) Potential problems could arise

- If a suitable condition is not mentioned for a series circuit. The rule may fire before all others when it encounters a star network.

- If all the delta networks are not converted to a star network initially, this may result in error value of the equivalent resistance.

c) For the system to be reduced to a circuit between two given nodes, a reference node should be identified.

d) Yes, the system can grow so as to encompass new components. By adding new slots for new components in the defined template.

APPENDIX

Note: circuit 1 and circuit 2 were run differently.

```
(deftemplate MAIN::R
    (slot Resistor)
    (slot Node_1)
    (slot Node_2 )
    (slot Res_value))

;;Assign values to each resistor
(deffacts circuit1
    (R (Resistor R1) (Node_1 A) (Node_2 B) (Res_value 2))
    (R (Resistor R2) (Node_1 B) (Node_2 C) (Res_value 2))
    (R (Resistor R3) (Node_1 B) (Node_2 D) (Res_value 4))
    (R (Resistor R4) (Node_1 C) (Node_2 D) (Res_value 3))
    (R (Resistor R5) (Node_1 C) (Node_2 E) (Res_value 4))
    (R (Resistor R6) (Node_1 D) (Node_2 E) (Res_value 2)))

(deffacts circuit2
(R (Resistor R1) (Node_1 A) (Node_2 B) (Res_value 2))
(R (Resistor R2) (Node_1 B) (Node_2 E) (Res_value 5))
(R (Resistor R3) (Node_1 B) (Node_2 D) (Res_value 1))
(R (Resistor R4) (Node_1 B) (Node_2 C) (Res_value 4))
(R (Resistor R5) (Node_1 C) (Node_2 D) (Res_value 4))
(R (Resistor R6) (Node_1 D) (Node_2 E) (Res_value 3))
(R (Resistor R7) (Node_1 A) (Node_2 C) (Res_value 2))
(R (Resistor R8) (Node_1 I) (Node_2 C) (Res_value 4))
(R (Resistor R9) (Node_1 L) (Node_2 H) (Res_value 3))
(R (Resistor R10) (Node_1 H) (Node_2 I) (Res_value 1))
(R (Resistor R11) (Node_1 J) (Node_2 H) (Res_value 5))
(R (Resistor R12) (Node_1 A) (Node_2 J) (Res_value 4))
(R (Resistor R13) (Node_1 J) (Node_2 I) (Res_value 1))
(R (Resistor R14) (Node_1 A) (Node_2 I) (Res_value 2))
(R (Resistor R15) (Node_1 A) (Node_2 K) (Res_value 4))
(R (Resistor R16) (Node_1 A) (Node_2 L) (Res_value 3))
(R (Resistor R17) (Node_1 A) (Node_2 K) (Res_value 1))
(R (Resistor R18) (Node_1 K) (Node_2 M) (Res_value 2))
(R (Resistor R19) (Node_1 L) (Node_2 I) (Res_value 4))
(R (Resistor R20) (Node_1 M) (Node_2 I) (Res_value 5)))

;DEFINE RULES
;;Series Reduction
(defrule series
    ?R1 <- (R
        (Resistor ?n1)
        (Node_1 ?Node_1)
        (Node_2 ?Node_2)
        (Res_value ?r1))
    ?R2 <- (R
        (Resistor ?n2)
        (Node_1 ?Node_2)
        (Node_2 ?node3)
        (Res_value ?r2))
    (not
        (or
            (R
                (Resistor ?n4&~?n2&~?n1)
```

```
                (Node_1 ?Node_2))
            (R
                (Resistor ?n4&~?n1&~?n2)
                (Node_2 ?Node_2))))
    =>
    (modify ?R1
        (Node_2 ?node3)
        (Res_value (+ ?r1 ?r2)))
    (retract ?R2))


;;Parallel Reduction
(defrule parallel
    ?R1 <- (R(Resistor ?n1)
        (Node_1 ?Node_1)
        (Node_2 ?Node_2)
        (Res_value ?r1))
    ?R2 <- (R(Resistor ?n2)
        (Node_1 ?Node_1)
        (Node_2 ?Node_2)
        (Res_value ?r2))
    (test (neq ?n1 ?n2))
    =>
    (modify ?R1
(Res_value
            (/ (* ?r1 ?r2)(+ ?r1 ?r2))))
    (retract ?R2))

;;Delta-Star Transformation
(defrule delta-star
    ?R1 <- (R(Resistor ?n1)
        (Node_1 ?Node_1)
        (Node_2 ?Node_2)
        (Res_value ?r1))
    ?R2 <- (R
        (Resistor ?n2)
        (Node_1 ?Node_2)
        (Node_2 ?node3)
        (Res_value ?r2))
    ?R3 <- (R
        (Resistor ?n3)
        (Node_1 ?Node_1)
        (Node_2 ?node3)
        (Res_value ?r3))
    (or
        (R
            (Resistor ?n4 & ~?n1&~?n2&~?n3)
            (Node_2 ?Node_2)))
    =>
    (modify ?R1
    (Node_2 (str-cat ?Node_1 ?Node_2))
        (Res_value (/ (* ?r1 ?r3) (+ ?r1 ?r2 ?r3))))
    (modify ?R2
        (Node_2 (str-cat ?Node_1 ?Node_2))
        (Node_1 ?Node_2)
        (Res_value (/ (* ?r1 ?r2) (+ ?r1 ?r2 ?r3))))
    (modify ?R3
        (Node_1 (str-cat ?Node_1 ?Node_2))
        (Res_value (/ (* ?r2 ?r3) (+ ?r1 ?r2 ?r3)))))
```

```
;;Equivalence Resistance
(defrule  Req
    ?R1 <- (R (Resistor ?n1)
        (Node_1 A)
        (Node_2 E)
        (Res_value ?r1))
    =>
    (printout t "The equivalent resistance, Req, for the circuit = " ?r1 "Ohms"))
(reset)
(run)
```