Circuit Language for the Intelligent Book

Intelligent Book Project
MIT Project for Mathematics and Computation
Cambridge Computer Laboratory
Summer 2003

One feature of the intelligent book is its ability to guide students in reasoning about the analysis and synthesis of electrical circuits. This capability is provided by a system of high-level teaching strategies, that are supported by low-level mechanisms for constraint propagation and truth maintenance. The circuit language bridges these two levels by providing means for describing electrical circuits and other information useful in implementing the teaching strategies. Beginning a description of a circuit, the circuit language system instantiates circuit models. These models are implemented as constraint networks that the teaching program uses for guiding dialogs about the analysis or synthesis of the circuit.

A single circuit diagram can be used to develop multiple models. For example, a single amplifier may have both a bias model and an incremental model. The circuit system will automatically generate all models that are called for, and appropriately interconnect them from a single circuit description.

In the circuit language, circuits are specified by wiring together primitive parts, and the circuit system automatically imposes the constraints that arise from the interconnection topology. A circuit specification includes the primitive parts and parameters together with additional relations among the circuit quantities. Some of these relations may relate quantities from different models. For example, the transconductance of a bipolar junction transistor is a quantity in the incremental model that is related to the collector current in the bias model.

In designing circuits, it is often necessary to make temporary assumptions. Preliminary reasoning about a design may involve simplifications that may be contradicted by more detailed reasoning at a later stage of design. For example, when reasoning about a transistor amplifier, it is often useful to initially assume that the β of the transistor is infinity and to later consider the consequences of the fact that β is actually finite. The circuit language provides means by which the teaching system may advance such an assumption and later retract it, as the student progresses through the design.

This memo describes the circuit language portion of the intelligent book. It does not describe the teaching system that uses the circuit language, and the underlying constraint mechanisms. These will be discussed elsewhere.

1 Circuit Elements

Every circuit element has terminals, parameters, and relations. A terminal has a current (describing current *into* the element from that terminal) and a potential. The relations may involve the terminal currents, the terminal potentials, and the parameters. Circuit elements may also include other circuit elements as parts and nodes to interconnect them.

The resistor shown in figure 1(a) is described as an elementary circuit element, which has no parts or nodes; it is described purely in terms of relations.

Figure 1(b) shows the circuit-language description of the resistor, defined using `part-type`. This definition specifies the type name `resistor`, the list of terminals `t1` and `t2`, and the single device parameter `resistance`.

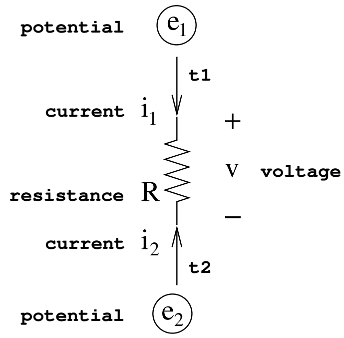
In general, a part is specified for a variety of models, such as *bias* or *incremental*. When a part is instantiated, the appropriate models for subparts are selected. Here, for the resistor, there is only one model, the special model called `any-model`, which specifies the behavior of the resistor when it is instantiated for any model. The `any-model` for the resistor has no subparts and no nodes for interconnecting them. It has one model-specific variable, called `voltage`.[1]

The resistor model stipulates three relations among various quantities. Each relation has an identifying name and an algebraic expression. The `kvl` relation says that the voltage across the resistor (from terminal `t1` to terminal `t2`) is the difference of the potentials at the two terminals, $v = e_1 - e_2$. The algebraic expression is expressed in Lisp notation, and uses a path-name notation, signaled by `>>`, to identify the quantities:

```
(= (>> voltage) (- (>> potential t1) (>> potential t2)))
```

The `kcl` and `ohm` relations are specified similarly. Finally, a part specification includes a list of cross-model relations. For a resistor there are no cross-model relations, so this list is empty.

[1] The difference between a model-specific quantity, such as the `voltage`, and a model-independent quantity, such as `resistance`, is that there is a separate instance of the model-specific quantity for each model made. For example, if a circuit containing a resistor is instantiated with a bias model and an incremental model, the `resistance` is shared among both models, but there is a separate bias voltage and incremental voltage.



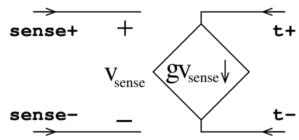

```
(part-type 'resistor                                ;type name
           '(t1 t2)                                 ;terminals
           '(resistance)                            ;part parameters
           '(;; list of models
             (any-model                             ;model name
              ()                                    ;model nodes
              ()                                    ;model parts
              (voltage)                             ;model variables
              (;; list of model-specific relations
               (kvl                                 ;relation name
                (= (>> voltage)
                   (- (>> potential t1)
                      (>> potential t2))))
               (kcl
                (= (>> current t2)
                   (- (>> current t1))))
               (ohm
                (= (>> voltage)
                   (* (>> resistance)
                      (>> current t1)))))))
           '(;; list of cross-model relations
             ))
```

Figure 1: (a) A resistor is a two-terminal device. Each terminal, `t1` and `t2`, has a `current` i_1 and i_2 and a `potential` e_1 and e_2. The `resistance` R of the resistor is a parameter of the device. The `voltage` v across the resistor is also indicated explicitly. (b) The resistor described in the circuit language.

```
(part-type 'voltage-source                    ;type name
           '(t1 t2)                           ;terminals
           '()                                ;part parameters
           '((any-model                       ;model name
              ()                              ;model nodes
              ()                              ;model parts
              (voltage)                       ;model variables
              ((kvl
                (= (>> voltage)
                   (- (>> potential t1) (>> potential t2))))
               (kcl
                (= (>> current t2) (- (>> current t1)))))))
           '())
```

Figure 2: Circuit language description of a voltage source.

Figure 2 shows the definition of an elementary voltage source part. This is a simple part similar to the resistor. It has one parameter, the `voltage` across the source. This is a model-specific parameter: the `voltage` of a source will different in the bias and incremental models. (For example, a DC source has zero incremental voltage.)




```
(part-type 'vccs
           '(sense+ sense- t+ t-)
           '(transconductance)
           '((any-model () ()
              (vsense vout)
              ((kvl-sense
                (= (>> vsense)
                   (- (>> potential sense+) (>> potential sense-))))
               (kcl-sense+
                (= (dimensioned 0 amperes) (>> current sense+)))
               (kcl-sense-
                (= (dimensioned 0 amperes) (>> current sense-)))
               (kvl-out
                (= (>> vout)
                   (- (>> potential t+) (>> potential t-))))
               (kcl-out
                (= (>> current t-) (- (>> current t+))))
               (control
                (= (>> current t+)
                   (* (>> transconductance) (>> vsense)))))))
           '())
```

Figure 3: A voltage-controlled current source.

Figure 3 shows a linear voltage-controlled current source together with its circuit-language description. This part has four terminals and one device parameter, the `transconductance` g. The voltage `vsense` between the two terminals `sense+` and `sense-` is related to the current of the dependent source (as measured going into the `t+` terminal) by the relation $i = gv_{\text{sense}}$, which is designated as the `control` relation in the part definition.

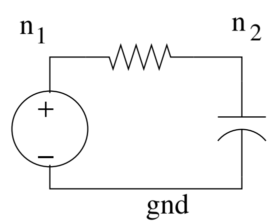
Figure 4 defines a capacitor as an elementary part. For the capacitor, we specify four models: a bias model, an incremental model, an impedance model, and a total model. The relations for these four models are different. For the bias model. the capacitor behaves as an open circuit; for the incremental model it behaves as a short circuit. The impedance model is defined in terms of complex impedances, and the total model involves derivatives. Observe that the voltages and currents in these models are all different, but the `capacitance`, which is specified as a parameter of the device, is shared among all four models.

```
(part-type 'capacitor
           '(t1 t2)
           '(capacitance)
           '((bias () ()
              (voltage)
              ((kvl
                (= (>> voltage)
                   (- (>> potential t1) (>> potential t2))))
               (kcl
                (= (>> current t2)
                   (- (>> current t1))))
               (opened
                (= (>> current t1)
                   (dimensioned 0 amperes)))))
             (incremental () ()
              (voltage)
              ((kvl
                (= (>> potential t1) (>> potential t2)))
               (kcl
                (= (>> current t2) (- (>> current t1))))
               (shorted
                (= (>> voltage)
                   (dimensioned 0 volts)))))
             (impedance () ()
              (voltage)
              ((kvl
                (= (>> voltage)
                   (- (>> potential t1) (>> potential t2))))
               (kcl
                (= (>> current t2)
                   (- (>> current t1))))
               (s-plane
                (= (>> current t1)
                   (* (* (>> capacitance) *s*)
                      (>> voltage))))))
             (total () ()
              (voltage)
              ((kvl
                (= (>> voltage)
                   (- (>> potential t1) (>> potential t2))))
               (kcl
                (= (>> current t2)
                   (- (>> current t1))))
               (capacitor
                (= (>> current t1)
                   (* (>> capacitance)
                      (derivative (>> voltage))))))))
           '())
```

Figure 4: Circuit language description of a capacitor, with four different models.

2 A compound circuit

Circuits can be defined as part types in terms of other parts. Figure 5 shows a simple series RC circuit, composed of a voltage source, a resistor, and a capacitor, together with the associated circuit-language description. For this circuit, the `any-model` declares three nodes and wires the parts among those nodes. The wiring diagram names the parts `v`, `r`, and `c`. Each part is specified by its part type together with the nodes in the circuit that the part terminals are attached to. In addition, there is one relation specified: that the potential of the node called `gnd` is zero. All other relations for the circuit come from the parts and from the wiring.




```
(part-type 'rc
           '()
           '()
           '((any-model
              (n1 n2 gnd)
              ((v voltage-source n1 gnd)
               (r resistor n1 n2)
               (c capacitor n2 gnd))
              ()
              ((gnd-potential
                (= (>> potential gnd) 0)))))
           '())
```

Figure 5: A simple RC circuit and its description in the circuit language.

3 Using the circuit system: a simple interaction

The `create-circuit` procedure instantiates circuit parts to produce a constraint network that supports interactive reasoning about the parts. We can manipulate the models by interacting directly with the circuit language in a read-eval-print loop; although, for the intelligent book, this manipulation is done indirectly, through the teaching part of the system.

Here's a simple example of interaction with the circuit language: We create bias and incremental models of the series RC circuit of figure 5:

```
(define n (create-circuit 'n 'rc '(bias incremental)))
```

Evaluating this expression binds `n` to an object in which one can set and examine the values of quantities, and perform constraint propagation.

We begin by postulating values for the resistance of `r` and for the voltage across the voltage source `v` in the bias model:[2]

```
(assume-value n '(resistance r) 2)
(assume-value n '(voltage v bias) 6)
```

Having set the voltage and the resistance, we now ask the constraint network to deduce what it can:

```
(propagate (constraint-network n))
```

If we now request the value of the voltage across the capacitor in the bias model, we get a reasonable answer, together with a brief explanation of how the value was deduced:[3]

```
(the-value n '(voltage c bias))
; ((voltage c bias) = 6)
;   (set by (-rhs:kvl c bias))
;     (because ((potential n2 bias) (potential gnd bias)))
```

The system has also deduced the voltage in the incremental model, which is zero, because the incremental capacitor is a short.

```
(the-value n '(voltage c incremental))
; ((voltage c incremental) = 0)
;   (set by (v:rhs:shorted c incremental))
;     (because ())
```

At this point, the value of the incremental voltage across the resistor is not yet available.

```
(the-value n '(voltage r incremental))
; ((voltage r incremental) is not assigned)
```

But if we make the further assumption that the incremental current into the top terminal of the capacitor is 1 mA, then more can be deduced.

[2]The resistance of a resistor is a model-independent parameter of the device, so we designate it without reference to a model. For the voltage source, `any-model` includes a quantity called `voltage`, and here we are designating the voltage in the bias model.

[3]The explanation shown here is only the final step of the deduction, which uses KVL for the capacitor in the bias model, together with the fact that the potential at `gnd` is 0 and the potential at node `n2` is 6. Finding the potential at `n2` required using the fact that there was no current through the capacitor (in the bias model), hence no current through the resistor (from KCL), hence no voltage drop across the resistor (from Ohm's Law). The circuit language includes procedures for examining such a chain of deductions. These are used by the teaching program in guiding students to reason about the circuit.

```
(assume-value n '(current t1 c incremental) .001)
(propagate (constraint-network n))
(the-value n '(voltage r incremental))
; ((voltage r incremental) = .002)
;   (set by (*rhs:ohm r incremental))
;     (because ((resistance r) (current t1 r incremental)))
```

In addition to propagation, the constraint propagation and truth maintenance systems underlying the circuit language can also uncover contradictions, and allow assumptions to be retracted. We'll see example of that below in section 5.

4 The transistor: A more complex part

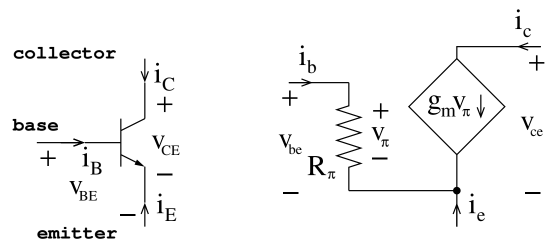


Figure 6: The NPN bipolar junction transistor and incremental model.

Figure 6 shows the bias and incremental models for an NPN bipolar junction transistor. This is a three-terminal device, whose definition as a part in the circuit language is shown in figure 7.

This definition exhibits some elements of the circuit language that are not illustrated by the simple parts described above. For the transistor, there are non-trivial bias and incremental models, with an intermodel relation. There are also alternative (inconsistent!) assumptions about the relations among the circuit quantities, which can be advanced or retracted under program control. We'll walk through the transistor definition for the transistor, step by step.

There are three terminals: `collector`, `emitter`, and `base`; and two device parameters: β and I_0. The part definition describes two models: bias model and incremental model.

The bias model has as parameters the base-emitter and collector-emitter voltages V_{BE} and V_{CE} and the KVL relations among these voltages and the potentials at the base, collector, and emitter terminals. We also have the KCL relation among the terminal currents. We'd have the same relations for any three-terminal device.

The transistor bias model also has a threshold voltage V_{T} and a saturation voltage V_{SAT} and relations that determine the device characteristics. Some of these relations, such as `beta-typical`, which specifies that $\beta = 100$, are assumed by the system to be always true, just as with all the examples of relations that we've seen so far.

```
(part-type 'npn-bjt                      ; type name
           '(collector base emitter)     ; terminals
           '(beta I0)                    ; global parameters
           '((bias
              ()                         ; model-specific nodes
              ()                         ; model-specific parts
              (vbe vce vthreshold vsat) ; model-specific parameters
              ((kvl-ce                   ; model-specific relations
                (= (>> vce)
                   (- (>> potential collector) (>> potential emitter))))
               (kvl-be
                (= (>> vbe)
                   (- (>> potential base) (>> potential emitter))))
               (kcl
                (= (+ (>> current collector) (>> current base))
                   (- (>> current emitter))))
               (operation
                (try
                 (cutoff
                  (= (>> current collector) (dimensioned 0 amperes)))
                 (amplifying
                  (and (> (>> current collector) (dimensioned 0 amperes))
                       (> (>> vce) (>> vsat))))
                 (switched-on
                  (and (> (>> current collector) (dimensioned 0 amperes))
                       (<= (>> vce) (>> vsat))))))
               (beta-heuristic
                (try
                 (beta-infinite
                  (= (>> current base) 0))
                 (beta-finite
                  (= (>> current collector)
                     (* (>> beta) (>> current base))))))
               (vbe-heuristic
                (try
                 (emitter-follows
                  (= (>> vbe) (>> vthreshold)))
                 (exponential
                  (= (>> current collector)
                     (* (>> I0) (- (exp (* q/kT (>> vbe))) 1))))))
               (beta-typical (= (>> beta) 100))
               (I0-typical (= (>> I0) (dimensioned 1e-12 amperes)))
               (vthreshold-typical
                (= (>> vthreshold) (dimensioned 0.6 volts)))
               (vsat-typical (= (>> vsat) (dimensioned 0.2 volts)))))
             (incremental
              ()                       ; model-specific nodes
              ((rpi resistor base   emitter) ; model-specific parts
               (source  vccs  base  emitter  collector emitter))
              ()                         ; model-specific parameters
              ((gm*rpi=beta              ; model-specific relations
                (= (* (>> transconductance source)
                      (>> resistance rpi))
                   (>> beta))))))
           '((gm=q/kT*IC                 ; intermodel relations
              (= (:> (transconductance source) incremental)
                 (* (:> (current collector) bias) q/kT))
              (bias incremental))))
```

Figure 7: The NPN bipolar junction transistor of figure 6 and its specification in the circuit language.

Other relations are in the form of alternative hypotheses, which can be proposed and retracted through interaction with the underlying truth-maintenance system. For example, we might first analyze a transistor circuit under the assumption of infinite β, where the base draws no current, and later do a more refined analysis with a finite value of β (for this model, with $\beta = 100$). The circuit language definition designates these choices by means of a `try` expression:

```
(beta-heuristic
 (try
  (beta-infinite
   (= (>> current base) 0))
  (beta-finite
   (= (>> current collector)
      (* (>> beta) (>> current base))))))
```

This specifies that there are alternatives: one where the base current is zero, and one where the collector current is equal to $\beta i_{\mathtt{B}}$. Interactions with the circuit model can specify which alternative to assume.[4]

There are also alternate `operation` assumptions about the region in which the transistor is operating: *cutoff*, where $I_{\mathrm{C}} = 0$; *amplifying*, where $I_{\mathrm{C}} > 0$ and $V_{\mathrm{CE}} > V_{\mathrm{SAT}}$, and *switched-on*, where $I_{\mathrm{C}} > 0$ and $V_{\mathrm{CE}} \leq V_{\mathrm{SAT}}$. Similarly, there is a `vbe-heuristic` choice of whether to assume that $V_{\mathrm{BE}} = V_{\mathrm{T}}$ (i.e., "the emitter follows the base"), or to use the more refined model $I_{\mathrm{C}} = I_0(e^{V_{\mathrm{BE}}q/kT} - 1)$.

The incremental model for the transistor (here we use the basic hybrid-π model) consists simply of a resistor r_π and a voltage-controlled current source whose transconductance is given by $g_m = \beta / r_\pi$.

Finally, there is a relation (called `gm=q/kT*IC`) in the transistor definition, between the quantities in the bias model and the quantities in the incremental model: the transconductance of g_m is equal to the collector bias voltage times q/kT. In general, a transistor design or analysis problem will require the system to reasoning with both models—just as we expect students to do.

Here, in the intermodel relation `gm=q/kT*IC`, we are using a different form of the path name mechanism, indicated by `:>` to refer to the incremental transconductance and the bias current. This is necessary here because the transistor may be one of many in a circuit. A model, say the bias model, of a circuit will have models for each of the transistors. Thus the path name for a model-specific variable of a transistor must have the transistor's name inserted before the model name. For example, if a circuit has two transistors, named `Q1` and `Q2` then the transconductance of a

[4]We'll see examples of this kind of control in the amplifier scenario below. Advancing and retracting various `try` assumptions can cause the truth-maintenance system to do significant reasoning. For example, if we tell the system to assume that the transistor is in the amplifying region and circuit analysis deduces that V_{CE} is less than V_{SAT}, then this leads to a contradiction, which the teaching program can bring to the attention of the student. This kind of reasoning with alternate assumptions is the kind of reasoning we would like students to go through in the analysis of simply transistor amplifiers: Assume that the transistor is amplifying, and choose circuit parameters to be consistent with this assumption.

transistor is named

```
(>> transconductance source Q1 incremental)
```

Notice that the transistor name `Q1` is interpolated in the path name.

5 A Common-Emitter Amplifier

Figure 8 shows a common-emitter amplifier circuit that consists of several parts: two voltage sources, two capacitors, four resistors, and an NPN bipolar junction transistor. This is the kind of circuit that we expect the intelligent book to be able to reason about with students.

The circuit-language definition for the amplifier is shown in figures 9 and 10. Notice that there are no terminals (i.e., the circuit is a complete circuit, with no places to connect to it from the outside). There are several parameters: the gain, the swing, the input impedance, and the minimum input impedance. The `any-model` for the circuit simply specifies the parts, and how they are wired together. Notice that we do not define separate bias and incremental models here, even though, as shown below, we will instantiate the amplifier with bias and incremental models. These will be constructed automatically from the `any-model` using the appropriate models for the parts: the bias model for the amplifier will contain bias models for the transistor and the sources, while in the incremental model for the amplifier will contain incremental models for the transistor and the sources.

In addition to the parts, there are several relations in the intermodel relations part of the description, including definitions of the swing and the input impedance, and stipulations that the ground potential is zero in both the bias and incremental models.[5]

Finally (figure 10) there are alternate rules with design heuristics for the gain of the amplifier. The simple rule of thumb is that the gain is $-R_{\mathrm{C}}/R_{\mathrm{E}}$ (provided that R_{E} is sufficiently large). A more delicate approximation incorporates a correction factor.[6]

5.1 Amplifier scenario

This section shows an interaction sequence with the amplifier model, as an illustration of the capabilities of the circuit language. This is not the same as an interaction

[5] Note that none of these relations are actually *inter-model* relations. Each one sits entirely within the bias model or the incremental model. In fact, we could have defined explicit bias and incremental models for the circuit, just for the purpose of holding these relations. But this would have required us to duplicate the information in the `any-model` description. In general, there is a lot of flexibility in how to specify these relations, and it's not always clear which way is best. Future versions of the language may revisit this.

[6] Explain this. Also explain what the estimator safety factor does.

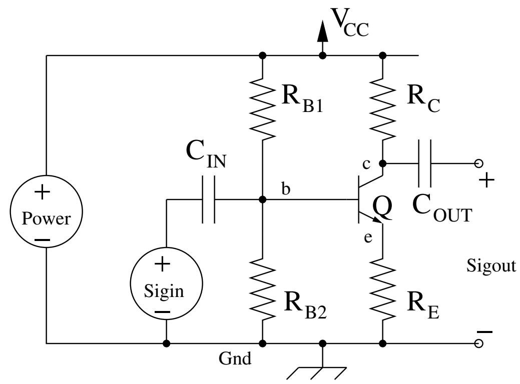


Figure 8: Common-emitter transistor amplifier circuit.

that a student would have with the intelligent book. Rather, it shows how the intelligent book teaching module might use the circuit language to support its interaction with a student.

We begin by instantiating the amplifier with bias and incremental models (and we also define the gain estimator parameter)

```
(define gain-estimator-safety-factor 10.0)
(define ce (create-circuit 'CE 'ce-amplifier '(bias incremental)))
```

Next, we set values for some of the circuit parameters: $V_{\mathrm{CC}} = 15$ Volts, gain equal to 10, swing 6 Volts, and minimum input impedance 10 kΩ. This use of the circuit language might, for example, be setting up a design exercise where the student is asked to find component values that meet these specifications:

```
(assume-value ce '(strength power) 15)
(assume-value ce '(gain) -10)
(assume-value ce '(swing) 6)
(assume-value ce '(min-input-impedance) 10000)
```

We'll also assume that some of the simple alternatives hold in determining the circuit behavior: the simple $R_{\mathrm{C}}/R_{\mathrm{E}}$ rule for the gain, that the transistor is operating in the amplifying region with the emitter voltage following the base, and that β for the transistor is infinite. These are rules of thumb that designers (and, we hope, our students) will use when beginning to design a circuit. Finally, we tell the system to deduce what it can from these assumptions and see that nothing interesting can be deduced yet:

```
(node-assume! (referent ce '(simple-estimate gain-heuristic)))
(node-assume! (referent ce '(amplifying operation q bias)))
(node-assume! (referent ce '(beta-infinite beta-heuristic q bias)))
(node-assume! (referent ce '(emitter-follows vbe-heuristic q bias)))
(propagate (constraint-network ce))
```

```
(part-type 'ce-amplifier                ; type-name
 '()                                    ; terminals
 '(gain swing input-impedance           ; global parameters
       min-input-impedance)
 '((any-model
    (vcc gnd c b e in out)              ; model-specific nodes
                                        ; model-specific parts
    ((power   dc-voltage-source   vcc gnd)
     (rb1     resistor            vcc b)
     (rb2     resistor            b   gnd)
     (rc      resistor            vcc c)
     (re      resistor            e   gnd)
     (q       npn-bjt             c   b  e)
     (cin     capacitor           in  b)
     (cout    capacitor           c   out)
     (sigin   voltage-signal-source  in  gnd)
     (sigout  open-circuit        out gnd))
    ()                                  ; model-specific parameters
    ((ground                            ; model-specific relations
      (= (>> potential gnd) 0))))
 '((input-impedance-definition           ; inter-model relations
    (= (/ (>> voltage sigin incremental) (>> current t1 cin incremental))
       (>> input-impedance))      ; models needed for this to be relevant
    (incremental))
   (min-impedance
    (> (>> input-impedance) (>> min-input-impedance))
    (incremental))
   (gain-definition
    (= (>> gain)
       (/ (>> voltage sigout incremental)
          (>> voltage sigin incremental)))
    (incremental))
   (swing-high
    (< (+ (>> swing)
          (- (>> potential c bias) (>> potential gnd bias)))
       (>> potential vcc bias))
    (bias))
   (swing-low
    (> (- (>> potential collector q bias) (>> potential emitter q bias))
       (+ (>> swing) (>> vsat q bias)))
    (bias))
```

Figure 9: Definition of a common-emitter transistor amplifier (continued in figure 10).

```
(gain-heuristic
 (try
  (simple-estimate
   (and (= (>> gain)
           (- (/ (>> resistance rc) (>> resistance re))))
        (> (>> resistance re)
           (* gain-estimator-safety-factor
              (/ 1
                 (* q/kT (>> current collector q bias)))))))
  (actual-gain
   (= (>> gain)
      (- (* (/ (>> resistance rc) (>> resistance re))
            (/ 1
               (+ 1
                  (/ 1 (>> beta q))
                  (/ 1
                     (* (>> transconductance source q incremental)
                        (>> resistance re)))))))))
 (bias incremental))))
```

Figure 10: Definition of a common-emitter transistor amplifier (continued from figure 9).

Now we assume that the bias potential on the collector is 10 volts and ask the system to make deductions. It finds a contradiction.

```
(assume-value ce '(potential c bias) 10)
(propagate (constraint-network ce))
; (Contradiction! #[uninterned-symbol 19 CEP94])
```

We can examine the reasoning steps that produced the contradiction. In this case, the collector voltage assumption violated the swing constraint.[7] The system can exhibit the chain of deductions that led to the contradiction:

```
(explain (car (cn-contradictions (constraint-network ce))))
;(CEP94 contradiction found by (<swing-high) (CEP93 CEP79))
;(CEP93 (v:lhs:swing-high) = 16 set by (+lhs:swing-high) (CEP68 CEP91))
;(CEP79 (potential vcc bias) = 15 set by (-rhs:kvl power bias) (CEP75 CEP35))
;(CEP68 (swing) = 6 set by assumption (CEP69))
;(CEP91 (v:1:lhs:swing-high) = 10 set by (-1:lhs:swing-high) (CEP89 CEP35))
;(CEP75 (voltage power bias) = 15 set by (=rhs:voltage-source power bias)
(CEP64))
;(CEP35 (potential gnd bias) = 0 set by (v:rhs:ground bias) ())
;(CEP89 (potential c bias) = 10 set by assumption (CEP90))
;(CEP64 (strength power) = 15 set by assumption (CEP65))
;Value: QED
```

It can also indicate the assumptions that gave rise to the contradiction:

[7]The problem here is that there are only 5 Volts between the 10-Volt collector potential and the 15-Volt V_{CC}, which does not leave room for a 6-Volt swing.

```
(support ce (car (cn-contradictions (constraint-network ce))))
;Value: ((swing) (potential c bias) (strength power))
```

The output printed here is not designed to be shown to students (or humans in general). It is included in this transcript to show the kind of information that is available to the teaching module.

To remove the contradiction, we retract the assumption about the bias potential being 10, and assume instead that it is 5 Volts. We also assume that the collector current is 0.01 Amps.

```
(retract-assumed-value ce '(potential c bias))
(propagate (constraint-network ce))
(assume-value ce '(potential c bias) 5)
(assume-value ce '(current collector q bias) 0.01)
```

Propagating these values yields another contradiction, arrived at through the long deductive chain shown in figure 11.[8]

We'll remove the contradiction by retracting the collector potential assumption, and postulating a new value of 8.5 Volts. Propagating with this value produces no contradiction:

```
(retract-assumed-value ce '(potential c bias))
(assume-value ce '(potential c bias) 8.5)
(propagate (constraint-network ce))
```

As part of its deductions, the system has found values for R_{C} and R_{E}. We can ask for these values, as well as for the assumptions from which, say, R_{C} was deduced:

```
(the-value ce '(resistance rc))
; ((resistance rc) = 650.)
;   (set by (*rhs:ohm rc bias))
;     (because ((voltage rc bias) (current t1 rc bias)))

(the-value ce '(resistance re))
; ((resistance re) = 65.)
;   (set by (/0:rhs:rhs:lhs:gain-heuristic))
;     (because ((resistance rc) (v:0:rhs:rhs:lhs:gain-heuristic)))

(support ce '(resistance rc))
;Value: ((strength power) (potential c bias) (current collector q bias))
```

In contrast, the system does not yet have a value for R_{B_1}.

[8] The contradiction arises because the difference between the collector potential and the emitter potential (which the system must compute), does not leave enough room for the swing, if the collector current and the emitter and collector resistances (again computed by the system) are to satisfy the conditions need to apply the simple gain heuristic. You can trace through the deductive chain to see how the system arrives at this conclusion.

```
(propagate (constraint-network ce))
; (Contradiction! #[uninterned-symbol 19 CEP125])

(explain (car (cn-contradictions (constraint-network ce))))
;(CEP125 contradiction found by (>swing-low) (CEP122 CEP72))
;(CEP122 (v:lhs:swing-low) = 4. set by (-lhs:swing-low) (CEP94 CEP119))
;(CEP72 (v:rhs:swing-low) = 6.2 set by (+rhs:swing-low) (CEP67 CEP26))
;(CEP94 (potential c bias) = 5 set by assumption (CEP95))
;(CEP119 (potential e bias) = 1. set by (-rhs:kvl re bias) (CEP118 CEP49))
;(CEP67 (swing) = 6 set by assumption (CEP68))
;(CEP26 (vsat q bias) = .2 set by (v:rhs:vsat-typical q bias) ())
;(CEP118 (voltage re bias) = 1. set by (*rhs:ohm re bias) (CEP110 CEP115))
;(CEP49 (potential gnd bias) = 0 set by (v:rhs:bias-ground) ())
;(CEP110 (resistance re) = 100. set by (/0:rhs:lhs:R:0:gain-heuristic) (CEP108
CEP84))
;(CEP115 (current t1 re bias) = .01 set by (kcl-node e bias) (CEP104))
;(CEP108 (resistance rc) = 1000. set by (*rhs:ohm rc bias) (CEP98 CEP105))
;(CEP84 (v:0:rhs:lhs:R:0:gain-heuristic) = 10 set by
(-rhs:lhs:R:0:gain-heuristic) (CEP73))
;(CEP104 (current emitter q bias) = -.01 set by (-rhs:kcl q bias) (CEP103))
;(CEP98 (voltage rc bias) = 10 set by (-rhs:kvl rc bias) (CEP75 CEP94))
;(CEP105 (current t1 rc bias) = .01 set by (-rhs:kcl rc bias) (CEP102))
;(CEP73 (v:rhs:lhs:R:0:gain-heuristic) = -10 set by (=lhs:R:0:gain-heuristic)
(CEP65 CEP53))
;(CEP103 (=kcl q bias) = .01 set by (+lhs:kcl q bias) (CEP96 CEP71))
;(CEP75 (potential vcc bias) = 15 set by (-rhs:kvl power bias) (CEP74 CEP49))
;(CEP102 (current t2 rc bias) = -.01 set by (kcl-node c bias) (CEP28 CEP96))
;(CEP65 (gain) = -10 set by assumption (CEP66))
;(CEP53 and-a1:R:0:gain-heuristic set by (and:R:0:gain-heuristic) (CEP51))
;(CEP96 (current collector q bias) = .01 set by assumption (CEP97))
;(CEP71 (current base q bias) = 0 set by (=R:0:beta-heuristic q bias) (CEP17
CEP16))
;(CEP74 (voltage power bias) = 15 set by (=rhs:voltage-source power bias)
(CEP63))
;(CEP28 (current t1 cout bias) = 0 set by (v:rhs:opened cout bias) ())
;(CEP51 T:0:gain-heuristic PREMISE)
;(CEP17 (v:rhs:R:0:beta-heuristic q bias) = 0 set by (c:rhs:R:0:beta-heuristic q
bias) ())
;(CEP16 T:0:beta-heuristic PREMISE)
;(CEP63 (strength power) = 15 set by assumption (CEP64))
;Value: QED

(pp (support ce (car (cn-contradictions (constraint-network ce)))))
;((swing)
; (beta-infinite beta-heuristic q bias)
; (simple-estimate gain-heuristic)
; (gain)
; (current collector q bias)
; (strength power)
; (potential c bias))
```

Figure 11: Chain of deductions supporting a contradiction elicited during the amplifier scenario.

```
(the-value ce '(resistance rb1))
;((resistance rb1) is not assigned)
```

But postulating a value for R_{B_2} determines R_{B_1}:

```
(assume-value ce '(resistance rb2) 20000)
(propagate (constraint-network ce))

(the-value ce '(resistance rb1))
; ((resistance rb1) = 220000.)
;   (set by (*rhs:ohm rb1 bias))
;     (because ((voltage rb1 bias) (current t1 rb1 bias)))
```

Now we'll assume an input signal strength of .01 Volts. This produces three separate contradictions:

```
(assume-value ce '(strength sigin) .01)

(propagate (constraint-network ce))
; (Contradiction! #[uninterned-symbol 19 CEP213])
; (Contradiction! #[uninterned-symbol 20 CEP212])
; (Contradiction! #[uninterned-symbol 21 CEP208])
```

We can examine the assumptions that led to each of the contradictions:

```
(pp (support ce (car (cn-contradictions (constraint-network ce)))))
;((min-input-impedance)
; (strength sigin)
; (resistance rb2)
; (emitter-follows vbe-heuristic q bias)
; (beta-infinite beta-heuristic q bias)
; (simple-estimate gain-heuristic)
; (gain)
; (current collector q bias)
; (strength power)
; (potential c bias))


(pp (support ce (cadr (cn-contradictions (constraint-network ce)))))
;((strength sigin)
; (resistance rb2)
; (emitter-follows vbe-heuristic q bias)
; (beta-infinite beta-heuristic q bias)
; (simple-estimate gain-heuristic)
; (gain)
; (current collector q bias)
; (strength power)
; (potential c bias))
```

```
(pp (support ce (caddr (cn-contradictions (constraint-network ce)))))
;((strength sigin)
; (simple-estimate gain-heuristic)
; (gain)
; (current collector q bias)
; (strength power)
; (potential c bias))
```

Let's try to remove the contradictions by relaxing the assumption that the amplifier gain is given by the simple rule of thumb. Unfortunately, we still get a contradiction:

```
(node-retract! (referent ce '(simple-estimate gain-heuristic)))

(propagate (constraint-network ce))
; (Contradiction! #[uninterned-symbol 30 CEP251])
(pp (support ce (car (cn-contradictions (constraint-network ce)))))
;((min-input-impedance)
; (resistance rb2)
; (emitter-follows vbe-heuristic q bias)
; (beta-infinite beta-heuristic q bias)
; (strength sigin)
; (gain)
; (current collector q bias)
; (strength power)
; (potential c bias))
```

The problem is that the input impedance is below the required 10kΩ minimum:

```
(the-value ce '(input-impedance))
; ((input-impedance) = 4801.578714450793)
;   (set by (/lhs:input-impedance-definition))
;     (because ((voltage sigin incremental) (current t1 cin
incremental)))
;Value: 4801.578714450793
```

Let's try a different value for R_{B_2}:

```
(retract-assumed-value ce '(resistance rb2))
(propagate (constraint-network ce))
(assume-value ce '(resistance rb2) 50000)

(propagate (constraint-network ce))
; (Contradiction! #[uninterned-symbol 32 CEP272])
```

This doesn't work either, but we do remove the contradiction if we also change our assumption about the collector bias current:

```
(retract-assumed-value ce '(current collector q bias))
(propagate (constraint-network ce))
(assume-value ce '(current collector q bias) 0.001)
(propagate (constraint-network ce))
```

Now that we appear to have values that meet the specs, we can examine the derived value of R_{E} and check the input impedance.

```
(the-value ce '(resistance re))
; ((resistance re) = 617.9681613101286)
;   (set by (*rhs:ohm re incremental))
;     (because ((voltage re incremental) (current t1 re incremental)))

(the-value ce '(input-impedance))
; ((input-impedance) = 26916.38761289355)
;   (set by (/lhs:input-impedance-definition))
;     (because ((voltage sigin incremental) (current t1 cin
incremental)))
```

This is well above the 10 kΩ minimum, and we can adjust the collector bias current to get a tighter match to the spec:

```
(retract-assumed-value ce '(current collector q bias))
(assume-value ce '(current collector q bias) 0.005)
(propagate (constraint-network ce))

(the-value ce '(input-impedance))
; ((input-impedance) = 10132.682461917972)
;   (set by (/lhs:input-impedance-definition))
;     (because ((voltage sigin incremental) (current t1 cin
incremental)))

(the-value ce '(resistance re))
; ((resistance re) = 123.59363226202572)
;   (set by (*rhs:ohm re incremental))
;     (because ((voltage re incremental) (current t1 re incremental)))
```

Now, with a "working" design, we'll remove the infinite β simplification and replace it with the more realistic $\beta = 100$:

```
(node-retract! (referent ce '(beta-infinite beta-heuristic q bias)))
(node-assume! (referent ce '(beta-finite beta-heuristic q bias)))

(propagate (constraint-network ce))
; (Contradiction! #[uninterned-symbol 33 CEP375])
```

Our design no longer meets the spec—the input impedance falls below 10 kΩ:

```
(the-value ce '(input-impedance))
; ((input-impedance) = 9772.319262809358)
;   (set by (/lhs:input-impedance-definition))
;     (because ((voltage sigin incremental) (current t1 cin
incremental)))
```

We can fix this by adjusting our assumed value for the collector current. Propagating now gives no contradiction:

```
(retract-assumed-value ce '(current collector q bias))
(assume-value ce '(current collector q bias) 0.003)
(propagate (constraint-network ce))
```

Finally, let's remove the simplifying assumption that the emitter follows the base, replacing it with the more accurate exponential model for the transistor. The design still works. (Whew!)

```
(node-retract! (referent ce '(emitter-follows vbe-heuristic q bias)))
(node-assume! (referent ce '(exponential vbe-heuristic q bias)))
(propagate (constraint-network ce))
```

We can examine the resistor values in the final working design:

```
(the-value ce '(input-impedance))
; ((input-impedance) = 14276.207105140602)
;   (set by (/lhs:input-impedance-definition))
;     (because ((voltage sigin incremental) (current t1 cin
incremental)))

(the-value ce '(resistance rb1))
; ((resistance rb1) = 256886.48085639544)
;   (set by (*rhs:ohm rb1 bias))
;     (because ((voltage rb1 bias) (current t1 rb1 bias)))

(the-value ce '(resistance rb2))
; ((resistance rb2) = 50000)
;   (set by assumption)

(the-value ce '(resistance rc))
; ((resistance rc) = 2166.6666666666665)
;   (set by (*rhs:ohm rc bias))
;     (because ((voltage rc bias) (current t1 rc bias)))

(the-value ce '(resistance re))
; ((resistance re) = 205.9893871033762)
;   (set by (*rhs:ohm re incremental))
;     (because ((voltage re incremental) (current t1 re incremental)))
```

We stress again that this long scenario is meant to illustrate the capability of the circuit language, and not to represent a proposed teaching interaction. For that, we need the teaching module, which uses the circuit language to support interactions guided by teaching strategies.