SpherePOP in Action: Scope, Reduction, and Admissible Replacement in Series-Parallel Networks

Flyxion
Independent Researcher

November 2026

Abstract

The reduction of a resistor network by successive series and parallel combination has the form of a rewriting procedure on nested regions: a bounded subnetwork is replaced by a simpler one that behaves equivalently at its boundary, and the procedure continues on the result. This essay asks in what precise sense that procedure is an instance of SpherePOP, a calculus in which nested bubbles are evaluated by local replacement, and it turns the suggestive resemblance into statements that can be tested. It separates three relations between an electrical network and a nested SpherePOP expression: shape resemblance, reduction image, and step bisimilarity, and it shows by explicit examples that none of the three implies another, while the translation of series-parallel networks into bubble terms, relative to a chosen decomposition, satisfies all of them and is a step-for-step correspondence. Series-parallel reduction is developed completely, with its rules, its preconditions, and the observations it preserves, and the preservation result is paired with an explicit account of what the reduced representation no longer exposes, namely internal currents, the distribution of dissipated power, and the construction history of the network. The Wheatstone bridge is given as a counterexample showing that not every circuit transformation is a nested-region replacement, and the star-delta transformation is examined as a proposed extension that requires regions with more than two boundary vertices. The essay then distinguishes canonical from permissible reduction sequences, counts the latter in a simple case, and proposes, as a reading to be checked against the calculus, that the operations of SpherePOP beyond evaluation, namely merging, collapsing, and refusing, are needed to say what a reduction means once its history is taken seriously.

1 Introduction: Nested Regions and Local Replacement

Spatial notations for nested expressions, in which an operator and its arguments are drawn inside a bounding region, have a long pedagogical use in teaching function composition. The same bounded-region picture arises in a different field when a resistor network is simplified. In a standard textbook treatment a complex combination circuit is reduced step by step: two resistors in series are replaced by a single resistor of their sum, two in parallel by a single resistor of their reciprocal-sum, and the process continues until one resistor remains, after which the values of the original components are recovered by working backward [7, 3]. One introductory text names the two phases the reduce and return approach, and it presents the networks as block diagrams in which each block may stand for any combination of elements and not only a single resistor [3]. Each step identifies a bounded subnetwork,

replaces it with an equivalent representation, and leaves the surrounding network available for further reduction. The resemblance to the evaluation of a nested expression, in which an innermost region is replaced by its value and evaluation continues outward, is evident. The question this essay addresses is what, precisely, that resemblance amounts to, and under what conditions a statement that the procedure is an instance of SpherePOP evaluation can be proved.

The thesis is that the relationship between a representation and the theory it is taken to instantiate should be examined at three distinct levels, and that confusion among them accounts for most of the apparent force of suggestive analogies. At the first level, a circuit diagram and a nested expression may share a shape: the nesting of series and parallel combinations in the circuit can be put in one-to-one correspondence with the nesting of bubbles in an expression. This is structural resemblance, and it is cheap to establish and cheap to exaggerate. At the second level, the replacement of a subnetwork by a single equivalent resistor may be a valid reduction, meaning that it preserves a specified set of observations at the boundary of the replaced region, whatever the rest of the network may be. This is a mathematical property with preconditions, and it can fail. At the third level, the individual replacement steps of the network may be matched one for one by the transitions of a rewriting system on bubble terms. This is a computational property, and it is not implied by the second. Strategies for choosing among available steps, and recorded histories of the steps taken, are additional structures that are studied after the matching of transitions has been established. The claim that series-parallel reduction is SpherePOP in action can be read at any of these levels, and it is most interesting, and most exposed to refutation, at the third.

The essay proceeds by making the claim as strong as the mathematics allows and then locating the point at which it stops being true. Section 2 introduces bubble terms, the `POP` rule, and the innermost-first strategy, and distinguishes reading a nested structure from acting on it. Section 3 defines series-parallel networks and the three correspondence relations, and proves the step-correspondence theorem. Section 4 proves the replacement property that makes series-parallel reduction valid and analyzes what it throws away. Section 5 gives the counterexamples, including the bridge circuit. Section 6 treats reduction as an executable history, distinguishing canonical from permissible sequences. Section 7 explains why SpherePOP cannot be identified with a graphical expression evaluator, using the operations `MERGE`, `COLLAPSE`, and `REFUSE`. Section 9 states what has and has not been established.

2 Bubble Terms and the Pop Rule

A bubble term is an expression drawn as a region, in the manner of the circles of evaluation used to teach function composition [16] and with the prefix notation of the Lisp family as its linear form [6]. Let $\mathbb{Q}_{>0}$ denote the positive rationals, and let $\Omega = \{+, \|\}$ be a set of operator symbols, where $+$ is ordinary addition and $\|$ is the reciprocal-sum operation $a_1 \,\|\, \cdots \,\|\, a_k = \left(\sum_i a_i^{-1}\right)^{-1}$, which for two arguments is $ab/(a+b)$. The restriction to these two operators is deliberate, since it is exactly what series-parallel reduction requires, and the question of how SpherePOP extends beyond it is deferred to Section 7.

Definition 2.1 (Bubble terms)**.** The set $\mathcal{B}$ of bubble terms is the least set such that every

$n \in \mathbb{Q}_{>0}$ is a bubble term, called a numeral, and whenever $\oplus \in \Omega$ and $t_1, \ldots, t_k \in \mathcal{B}$ with $k \geq 2$, the expression $(\oplus\ t_1\ \cdots\ t_k)$ is a bubble term. A term of the second form is a bubble with operator $\oplus$ and children $t_1, \ldots, t_k$. Its drawing is a region labeled $\oplus$ that spatially contains the drawings of its children.

Three notions that are easily conflated should be separated at the outset. Syntactic containment is the relation that holds between a bubble and the bubbles drawn inside it, and it is a property of the drawing. Semantic scope is the relation that determines which subterms contribute to the value of which bubble, and for bubble terms it coincides with containment, although for richer languages with binding it need not. Evaluation order is the order in which bubbles are resolved, and containment constrains it without selecting a unique total order, because two bubbles that do not contain one another may be resolved in either order, while a bubble containing another must in general wait for it. A spatial drawing makes the first of these visible and invites the assumption that it has settled the other two. It has not, and the point at which that assumption is made explicit is the point at which a notation becomes a programming language.

The value of a term is given by structural recursion: $[\![n]\!] = n$ and $[\![(\oplus\ t_1 \cdots t_k)]\!] = \oplus([\![t_1]\!], \ldots, [\![t_k]\!])$. Reading a term is a traversal that serializes the spatial structure into linear notation, and it does not change the term. A parser in an assembler similarly exposes the components of a command without executing it [12]. Acting on a term is different in kind.

Definition 2.2 (The `POP` rule)**.** A bubble is *ready* if all of its children are numerals. The rule `POP` replaces a ready bubble $(\oplus\ n_1 \cdots n_k)$ by the numeral $\oplus(n_1, \ldots, n_k)$, and it may be applied to any ready bubble occurring anywhere in a term, as a rewrite of that subterm in its context. The relation $t \longrightarrow t'$ holds when t' results from t by one application of `POP`. Step granularity is a parameter of the correspondence studied below: the all-children rule is used throughout, except for the finer-grained aggregation rule of Section 6, and under it a flat series bubble with numeral children reduces in a single step.

The rule is local, in that it inspects only a bubble and its immediate children, and it is destructive, in that the bubble and its children are replaced by a single numeral. The customary instruction, to find the innermost bubble and pop it, is a strategy for choosing among ready bubbles. In a term every ready bubble is innermost in the sense that it contains no further bubble, so the instruction leaves a choice among several ready bubbles unresolved and the usual tie-breaking convention is to take the leftmost. This is the sense in which a rule and a strategy are different objects, and the distinction returns in Section 6.

Lemma 2.3. *Every non-numeral bubble term contains a ready bubble. Consequently the irreducible terms are exactly the numerals, every maximal sequence of `POP` steps from t ends in a numeral, and that numeral equals $[\![t]\!]$.*

Proof. A non-numeral term has at least one bubble, and a bubble of maximal depth has only numerals as children, so it is ready. Each `POP` step preserves $[\![\cdot]\!]$ by the recursive definition of the value, and it decreases the number of bubbles by one, so every sequence terminates, and the terminal term is a numeral whose value is $[\![t]\!]$. □

Lemma 2.3 shows that the final answer of any sequence of pops is independent of the choices made along the way. It says nothing about the intermediate terms, which differ,

and this asymmetry between an invariant result and a variable history is the structural fact on which the later sections turn.

3 Networks and Three Levels of Correspondence

Consider two-terminal resistor networks built from single resistors by series and parallel composition. A two-terminal network is a network together with a designated ordered pair of terminals.

Definition 3.1 (Series-parallel network terms)**.** The set $\mathcal{N}$ of series-parallel network terms is the least set such that $\mathsf{R}(r) \in \mathcal{N}$ for each $r \in \mathbb{Q}_{>0}$, and $\mathsf{S}(N_1, \ldots, N_k) \in \mathcal{N}$ and $\mathsf{P}(N_1, \ldots, N_k) \in \mathcal{N}$ whenever $k \geq 2$ and $N_i \in \mathcal{N}$. The term $\mathsf{R}(r)$ denotes a single resistor of resistance r between the terminals; $\mathsf{S}(N_1, \ldots, N_k)$ denotes the chain in which the second terminal of N_i is joined to the first terminal of N_{i+1}; and $\mathsf{P}(N_1, \ldots, N_k)$ denotes the network in which all first terminals are joined and all second terminals are joined.

Each such term determines a graph and each graph carries the electrical behavior imposed by Ohm's law on its edges and Kirchhoff's laws [9] at its nodes. The terminal law of a network N is the set $\mathcal{B}(N) \subseteq \mathbb{R}^2$ of pairs (V, I), a voltage across the terminals and a current entering the first terminal, that are realized by some assignment of node voltages and edge currents satisfying all the laws. For a network of positive resistors the node voltages are determined by the terminal voltage, and the terminal law is a line through the origin, $\mathcal{B}(N) = \{(V, I) : V = \rho(N)\, I\}$, where the terminal resistance $\rho(N)$ is computed by $\rho(\mathsf{R}(r)) = r$, $\rho(\mathsf{S}(N_1, \ldots, N_k)) = \sum_i \rho(N_i)$, and $\rho(\mathsf{P}(N_1, \ldots, N_k)) = \rho(N_1) \parallel \cdots \parallel \rho(N_k)$. This is the familiar recipe, and it is also a structural recursion with the same shape as the one defining $[\![\cdot]\!]$.

The translation $\tau : \mathcal{N} \to \mathcal{B}$ makes the resemblance explicit: $\tau(\mathsf{R}(r)) = r$, $\tau(\mathsf{S}(N_1, \ldots, N_k)) = (+\ \tau(N_1) \cdots \tau(N_k))$, and $\tau(\mathsf{P}(N_1, \ldots, N_k)) = (\parallel\ \tau(N_1) \cdots \tau(N_k))$. The general statement that $\rho(N) = [\![\tau(N)]\!]$ is an immediate induction. It would be a mistake to read this as the whole of the claim that series-parallel reduction is SpherePOP evaluation, and the three relations that follow are intended to prevent that mistake.

Definition 3.2 (Network steps)**.** A *network step* $N \Rightarrow N'$ replaces an occurrence of a subterm $\mathsf{S}(\mathsf{R}(a_1), \ldots, \mathsf{R}(a_k))$ by $\mathsf{R}(a_1 + \cdots + a_k)$, or an occurrence of a subterm $\mathsf{P}(\mathsf{R}(a_1), \ldots, \mathsf{R}(a_k))$ by $\mathsf{R}(a_1 \parallel \cdots \parallel a_k)$, anywhere inside N.

Definition 3.3 (Three relations between a network and a bubble term)**.** Let N be a network term and t a bubble term.

(L1) Shape resemblance. N and t *resemble in shape* if the unlabeled rooted tree of the decomposition of N is isomorphic, with children unordered, to the unlabeled rooted tree of t. Operator labels and leaf values play no role.

(L2) Reduction image. A *reading* of t in N is an assignment φ of a subnetwork $\varphi(s)$ of N to every subterm occurrence s of t, with subnetworks taken modulo associativity of series and parallel composition and commutativity of parallel composition, series composition being order-preserving along the chain, such that $\varphi(t) = N$; if $s = (+\ s_1 \cdots s_k)$ then $\varphi(s)$ is the series composition of $\varphi(s_1), \ldots, \varphi(s_k)$ in this order along the chain, and if $s = (\parallel\ s_1 \cdots s_k)$ then $\varphi(s)$ is their parallel composition; and if s is a

numeral n then $\rho(\varphi(s)) = n$. The term t is a *reduction image* of N if it has a reading in N.

(L3) Step bisimilarity. A relation $\mathcal{R} \subseteq \mathcal{N} \times \mathcal{B}$ is a *step bisimulation* if $(N, t) \in \mathcal{R}$ implies $[\![t]\!] = \rho(N)$, and moreover (i) whenever $N \Rightarrow N'$ there is t' with $t \longrightarrow t'$ and $(N', t') \in \mathcal{R}$, and (ii) whenever $t \longrightarrow t'$ there is N' with $N \Rightarrow N'$ and $(N', t') \in \mathcal{R}$. The pair (N, t) satisfies (L3) if it belongs to some step bisimulation. Thus (L1), (L2), and (L3) are all predicates on pairs, and the definition of a step bisimulation is the criterion that makes a relation admissible as a witness for (L3).

Remark 3.4 (Identity of parts). A part $\varphi(s)$ is identified with a set of resistor occurrences of N, so that two parts are the same when they contain the same resistors. Reading modulo associativity means that regrouping adjacent children of a series chain, or children of a parallel composition, is allowed, and commutativity is allowed only for parallel composition. The restriction on series composition matters. If series children could be reordered freely, a reading could group nonadjacent resistors, such as the resistors of values 1 and 4 in $\mathsf{S}(\mathsf{R}(1), \mathsf{R}(2), \mathsf{R}(4))$, whose union is separated by the resistor of value 2 and communicates with the rest of the network through four boundary vertices, so that it is not a two-terminal-isolated part and Proposition 4.2 would not apply to it. Reordering a series chain is an electrically equivalent rearrangement, since it changes no resistor current or voltage drop, but it is not a replacement of a region of the original graph, and it is not part of a reading. The block-diagram presentation of series-parallel networks in introductory texts, in which a block may be a combination of elements [3], is the informal counterpart of a reading whose leaves are parts and not single resistors.

The three relations are logically independent in the following sense: none of them implies either of the others. The relation (L3) is a deliberately coarse operational relation, chosen so that its non-implications hold under minimal assumptions, and the position-preserving correspondence that carries the content is Theorem 3.6. The six non-implications are witnessed by three examples, each satisfying exactly one relation.

Only shape resemblance. Take $N = \mathsf{S}(\mathsf{R}(1), \mathsf{R}(1))$ and $t = (\parallel\ 1\ 1)$. The trees are isomorphic as unlabeled trees, so (L1) holds. No reading exists, because a reading would make N the parallel composition of two subnetworks while the decomposition of N has a series root, so (L2) fails. The values are $\rho(N) = 2$ and $[\![t]\!] = \frac{1}{2}$, so no step bisimulation can contain the pair and (L3) fails. A resemblance of shape says nothing about what the operators mean, and an interpretation that swaps the roles of the two operators preserves every shape feature while destroying the correspondence. This shows that (L1) implies neither (L2) nor (L3), and it is the sense in which the first relation is the cheapest to establish. Requiring operator labels and leaf values to match, preserving the order of series children and permitting permutation only of parallel children, gives the direct syntactic correspondence underlying τ, which is the strongest of the three.

Only a reduction image. Take $N = \mathsf{S}(\mathsf{R}(2), \mathsf{R}(2))$ and the numeral $t = 4$. The assignment $\varphi(t) = N$ is a reading, since $\rho(N) = 4$, so (L2) holds. The shapes differ, a root with two leaves against a single leaf, so (L1) fails. And N admits the step $N \Rightarrow \mathsf{R}(4)$ while t admits no step, so no step bisimulation relates them and (L3) fails. This shows that (L2) implies neither (L1) nor (L3), and in particular that a reduction image may skip arbitrarily many steps, which (L3) forbids.

Only step bisimilarity. Take $N = \mathsf{S}(\mathsf{R}(1), \mathsf{R}(1))$ and $t = (+\ 1\ \frac{1}{2}\ \frac{1}{2})$. Both have value 2 and each admits exactly one step, to a single resistor or numeral of value 2, so $\mathcal{R} = \{(N, t), (\mathsf{R}(2), 2)\}$ is a step bisimulation and (L3) holds. The shapes differ, a root with two leaves against a root with three, so (L1) fails. No reading exists, because the numeral $\frac{1}{2}$ would have to be assigned a subnetwork of N with that resistance and N has none, so (L2) fails. The same device with $t = (\|\ 4\ 4)$ gives a pair that has the same shape as N and satisfies (L3) but not (L2). This shows that (L3) implies neither (L1) nor (L2), and it shows that step bisimilarity as defined is a weak condition: a witnessing relation can pair states by their values and their available transitions without preserving resistor identities, operator labels, or positions.

Equality of values is necessary but not sufficient for (L2) and (L3), and it is incomparable with (L1). The example of the first kind is $N = \mathsf{S}(\mathsf{R}(2), \mathsf{R}(2))$ and $t = (+\ 1\ 3)$. Here $[\![t]\!] = 4 = \rho(N)$, and the pair even satisfies (L1) and (L3), since $\{(N, t), (\mathsf{R}(4), 4)\}$ is a step bisimulation, but no reading exists, because the numerals 1 and 3 would have to be assigned subnetworks of N with those resistances, and the only parts of N are the two resistors of value 2 and N itself. The values agree globally and the leaf-level correspondence fails. Insufficiency for (L3) is shown by $N = \mathsf{S}(\mathsf{R}(2), \mathsf{R}(2))$ and the numeral $t = 4$, which have equal values and no step bisimulation. The first counterexample of this section shows that (L1) does not imply equality of values. The translation τ is the case in which all three relations hold at once, with the strongest form of each.

The weakness of (L3) can be stated exactly. Let $b(N)$ be the number of series and parallel constructor occurrences in N and $b(t)$ the number of bubbles in t.

Proposition 3.5 (What step bisimilarity detects)**.** *For network terms and bubble terms with the all-children rules, (N, t) satisfies (L3) if and only if $\rho(N) = [\![t]\!]$ and $b(N) = b(t)$.*

Proof. Each network step and each `POP` step decreases the relevant count by exactly one, and a term with count zero is a single resistor or numeral and admits no step. Suppose (N, t) lies in a step bisimulation $\mathcal{R}$. Then $\rho(N) = [\![t]\!]$. Following $b(N)$ steps of N to a single resistor and matching each by clause (i), we reach $(\mathsf{R}(r), t')$ in $\mathcal{R}$ with t' reached by $b(N)$ steps from t; clause (ii) shows t' admits no step, so t' is a numeral and $b(t) = b(N)$. Conversely let $\mathcal{R}'$ be the set of pairs with equal value and equal count. If $(N', t') \in \mathcal{R}'$ and $N' \Rightarrow N''$, then the count of t' is positive, so by Lemma 2.3 it contains a ready bubble whose `POP` gives t'' with the same value and count one less, and $(N'', t'') \in \mathcal{R}'$. The argument for clause (ii) is symmetric. □

Step bisimilarity therefore detects the value and the length of a complete execution, and it ignores branching and dependency structure. This does not invalidate the definition, but it sharpens the earlier remark that (L3) alone is a weak condition: it is the correspondence of Theorem 3.6, with its positions and operator labels, that carries the content.

Theorem 3.6 (Step correspondence)**.** *The relation $\mathcal{R}_\tau = \{(N, \tau(N)) : N \in \mathcal{N}\}$ is a step bisimulation. More precisely, for every N and every ready bubble of $\tau(N)$ there is exactly one network step of N in the same position, and conversely, and the results again correspond under τ: $N \Rightarrow N'$ if and only if $\tau(N) \longrightarrow \tau(N')$ by the corresponding step, and the pair is witnessed by the reading that assigns to each subterm of $\tau(N)$ the subnetwork of N that τ translates to it.*

Proof. The map τ is compositional, and $\tau(M)$ is a numeral exactly when M is a single resistor $\mathsf{R}(r)$, in which case $\tau(M) = r$. Consequently a subterm of $\tau(N)$ is a ready bubble $(+\ n_1 \cdots n_k)$ exactly when the corresponding subterm of N is $\mathsf{S}(\mathsf{R}(n_1), \ldots, \mathsf{R}(n_k))$, and likewise for $\|$ and P. These are the redexes of Definition 3.2. Replacing the network redex by $\mathsf{R}(c)$ with $c = n_1 + \cdots + n_k$ (respectively $n_1 \,\|\, \cdots \,\|\, n_k$) and applying τ gives the term obtained from $\tau(N)$ by replacing the ready bubble with the numeral c, which is the result of `POP`. The same computation read backwards gives the converse. The value condition $[\![\tau(N)]\!] = \rho(N)$ is the induction noted above, and the identity assignment on subterms is a reading. □

Theorem 3.6 is the precise form of the claim that SpherePOP in Action is a correct description of the textbook procedure, with a stated scope: for a chosen decomposition term and the stipulated all-children replacement rules, the nested calculus and the replacement procedure are the same transition system under τ, step for step. The theorem is in effect an isomorphism of two term-rewriting systems whose constructors and rules were chosen to match. It does not by itself establish that arbitrary graph-level series-parallel reductions correspond to the reductions of a fixed term. For the term $\mathsf{S}(\mathsf{R}(a), \mathsf{S}(\mathsf{R}(b), \mathsf{R}(c)))$ the inner reduction of b and c is available, whereas combining the adjacent resistors a and b of the underlying chain is not available without reassociation, and the flat term $\mathsf{S}(\mathsf{R}(a), \mathsf{R}(b), \mathsf{R}(c))$ reduces entirely in one step under the all-children rule. The graph, the chosen decomposition, and the granularity of the permitted rewrites therefore determine different transition systems, and a graph-level correspondence requires structural congruence, as the `MERGE` of Section 7, or additional rewriting rules. Proposition 4.3 in the next section shows that every reading, and in particular the reading supplied by the theorem, licenses each replacement as a valid reduction.

A comparison with the translation performed by an assembler clarifies what the theorem requires. An assembler translates each instruction of a symbolic program into its binary encoding, directly, one to one, and separately for each command [12]. Such a translation is a transformation of representations, and it preserves the intended machine behavior of each instruction across a change of encoding. It does not perform the computation that the instruction specifies. A `POP`, by contrast, evaluates a ready subterm and replaces it with its value, so that it preserves the value of an expression while advancing its evaluation. The one-to-one character of the assembler's translation therefore would not, taken by itself, establish anything like Theorem 3.6, which requires that reduction transitions correspond and not merely that constructors or commands be matched by a dictionary. The assembler therefore illustrates why a correspondence between representations must be distinguished from an operational correspondence: command-by-command encoding alone supplies no proof that transitions of specified execution systems correspond.

4 Valid Reduction and What It Discards

The central fact that makes series-parallel reduction a legitimate computation, and not merely a pleasant coincidence of notation, is a compositionality property. A subnetwork can be replaced by anything with the same terminal law because the rest of the network can observe the subnetwork only through its terminals. This is a statement about contexts, in the sense in which contextual equivalence is used to justify local replacement in programming

languages [13], and it is the electrical analogue of the congruence property that distinguishes a sound rewriting rule from a mere coincidence of values [14, 2].

Definition 4.1. Let M be a two-terminal subnetwork of a network N. M is *two-terminal-isolated* in N if every node of M other than its two terminals is incident only to edges of M. A *context* $C[\,\cdot\,]$ is a network with a distinguished two-terminal hole, and $C[M]$ denotes the network obtained by filling the hole with M. Two two-terminal networks are *observationally equivalent*, written $M \approx M'$, if $\mathcal{B}(M) = \mathcal{B}(M')$.

Proposition 4.2 (Replacement)**.** *If $M \approx M'$ then $\mathcal{B}(C[M]) = \mathcal{B}(C[M'])$ for every context C. In particular, for series-parallel networks, replacing any subterm M by $\mathsf{R}(\rho(M))$ does not change ρ of the whole network, and does not change the terminal law of any enclosing subnetwork.*

Proof. Write the network equations of $C[M]$ as the union of the equations of C and the equations of M, linked by the voltage V_M across the hole and the current I_M entering it. The internal nodes of M are incident only to edges of M, so the equations of M constrain the pair (V_M, I_M) and the internal unknowns of M, and nothing else. Eliminating the internal unknowns of M leaves the condition $(V_M, I_M) \in \mathcal{B}(M)$ as the only contribution of M to the remaining system. Since $\mathcal{B}(M) = \mathcal{B}(M')$, the remaining system is the same for $C[M]$ and $C[M']$, so the projections of their solution sets onto the variables of C coincide, and in particular so do their terminal laws. The second statement follows because $\mathcal{B}(M) = \{(V, I) : V = \rho(M)I\} = \mathcal{B}(\mathsf{R}(\rho(M)))$. □

Proposition 4.3 (Readings are valid)**.** *If φ is a reading of t in N, then $\rho(\varphi(s)) = [\![s]\!]$ for every subterm s of t, in particular $[\![t]\!] = \rho(N)$, and replacing the part $\varphi(s)$ by $\mathsf{R}([\![s]\!])$ yields a network N' with $\mathcal{B}(N') = \mathcal{B}(N)$.*

Proof. The first claim is an induction on s using the clauses of a reading and the recursive definitions of ρ and $[\![\cdot]\!]$. Each part $\varphi(s)$ is a node of a series-parallel decomposition of N and is therefore two-terminal-isolated, since every node internal to a series-parallel component is incident only to edges of that component. By the first claim $\mathcal{B}(\varphi(s)) = \mathcal{B}(\mathsf{R}([\![s]\!]))$, and Proposition 4.2 applies. □

The proposition has a precondition that is easy to overlook in a picture, namely two-terminal isolation. In a drawn circuit a bubble drawn around a subnetwork is a closed curve, and the proposition applies only if the region communicates with its context through exactly two identified boundary vertices. Several wire segments may meet the same boundary vertex, so the number of segments crossing the curve is not the criterion. If a third boundary vertex exists, because an internal node is tapped by a voltmeter, a supply, or another branch, then the region has three boundary vertices and Proposition 4.2 does not apply to it. The graphical device of a bubble does not enforce this condition; the boundary vertices through which the region communicates must be identified. This is the first place where the correspondence needs a rule that the bare circles-of-evaluation notation does not provide, because in a nested expression every region has exactly one output by construction.

Example 4.4 (The textbook network)**.** The network of the textbook figure has nine resistors, $R_1 = 360$, $R_2 = 300$, $R_3 = 270$, $R_4 = 330$, $R_5 = 430$, $R_6 = 450$, $R_7 = 510$,

$R_8 = 750$, and $R_9 = 240$ ohms, driven by a 60 V source. In the notation of Section 3 it is

$$N_0 = \mathsf{S}\Big(\mathsf{R}(360),\ \mathsf{P}\big(\mathsf{S}(\mathsf{R}(300),\ \mathsf{P}(\mathsf{S}(\mathsf{R}(270),\mathsf{R}(330)),\ \mathsf{R}(450)),\ \mathsf{R}(430)),\ \mathsf{S}(\mathsf{R}(510),\mathsf{R}(750))\big),\ \mathsf{R}(240)\Big),$$

and its image is

$$\tau(N_0) = (+\ 360\ (\|\ (+\ 300\ (\|\ (+\ 270\ 330)\ 450)\ 430)\ (+\ 510\ 750))\ 240).$$

The textbook proceeds in six steps. It forms $R_{c1} = R_3 + R_4 = 600$, then $R_{c2} = R_{c1} \parallel R_6 = 1800/7 \approx 257.143$, then $R_{c3} = R_2 + R_{c2} + R_5 \approx 987.143$, then $R_{c4} = R_7 + R_8 = 1260$, then $R_{c5} = R_{c3} \parallel R_{c4} \approx 553.503$, and finally $R_T = R_1 + R_{c5} + R_9 \approx 1153.503$. Each step is one application of `POP` to a ready bubble of $\tau(N_0)$, and Figure 1 shows the network, with the first ready bubble enclosed and boundary vertices marked, the network after steps a through d, and the final resistor; and each satisfies the hypotheses of Proposition 4.2. The exact value is $R_T = 1814460/1573$, so the printed figures are roundings. The textbook then states the principle on which Proposition 4.6 rests, that once the values of the simple circuit are found they can be placed back in the original circuit to find the other values [7].

The equivalence is relative to a choice of observation, and the choice is what the reduction preserves. The terminal law is preserved, and with it the total power VI delivered to the network, since the power is a function of the terminal pair. In general the following quantities are not exposed by the reduced network, which does not determine them, although the original circuit is physically unchanged and particular quantities may be recoverable in particular cases. The current in each original resistor, the voltage across each, and the power dissipated in each, are lost when the subnetwork is replaced. The identity of the construction, which resistors were combined in what order, is lost as well.

Proposition 4.5 (Loss)**.** *There exist networks $M \approx M'$ whose resistors carry different currents for the same terminal pair, and networks $M \approx M'$ whose resistors carry the same current but experience different voltage drops and dissipate power differently, so that the reduced network does not in general determine the internal state of the original.*

Proof. For currents take $M = \mathsf{R}(600)$ and $M' = \mathsf{P}(\mathsf{R}(1200), \mathsf{R}(1200))$. Both have terminal resistance 600, so $M \approx M'$. Under a nonzero terminal current I the single resistor of M carries I, whereas each resistor of M' carries $I/2$. For voltages and power take $M = \mathsf{S}(\mathsf{R}(270), \mathsf{R}(330))$ and $M' = \mathsf{S}(\mathsf{R}(300), \mathsf{R}(300))$. Both have terminal resistance 600. Every resistor in either chain carries the terminal current I, so the currents agree, but the voltage drops across the resistors of M are $270I$ and $330I$ and those of M' are $300I$ each, and the dissipated powers $270I^2, 330I^2$ differ from $300I^2, 300I^2$ although the sums agree. The common reduced network $\mathsf{R}(600)$ cannot distinguish any of these cases. □

The loss is not total, because a reduction that retains its log can be run in reverse. Suppose each application of `POP` is recorded in the log together with the subterm that it replaced. Then, given the terminal pair of the full network, the internal state can be reconstructed by a downward pass: at a series node the same current flows through every child and the voltage divides in proportion to the child's resistance, and at a parallel node the same voltage appears across every child and the current divides in inverse proportion.

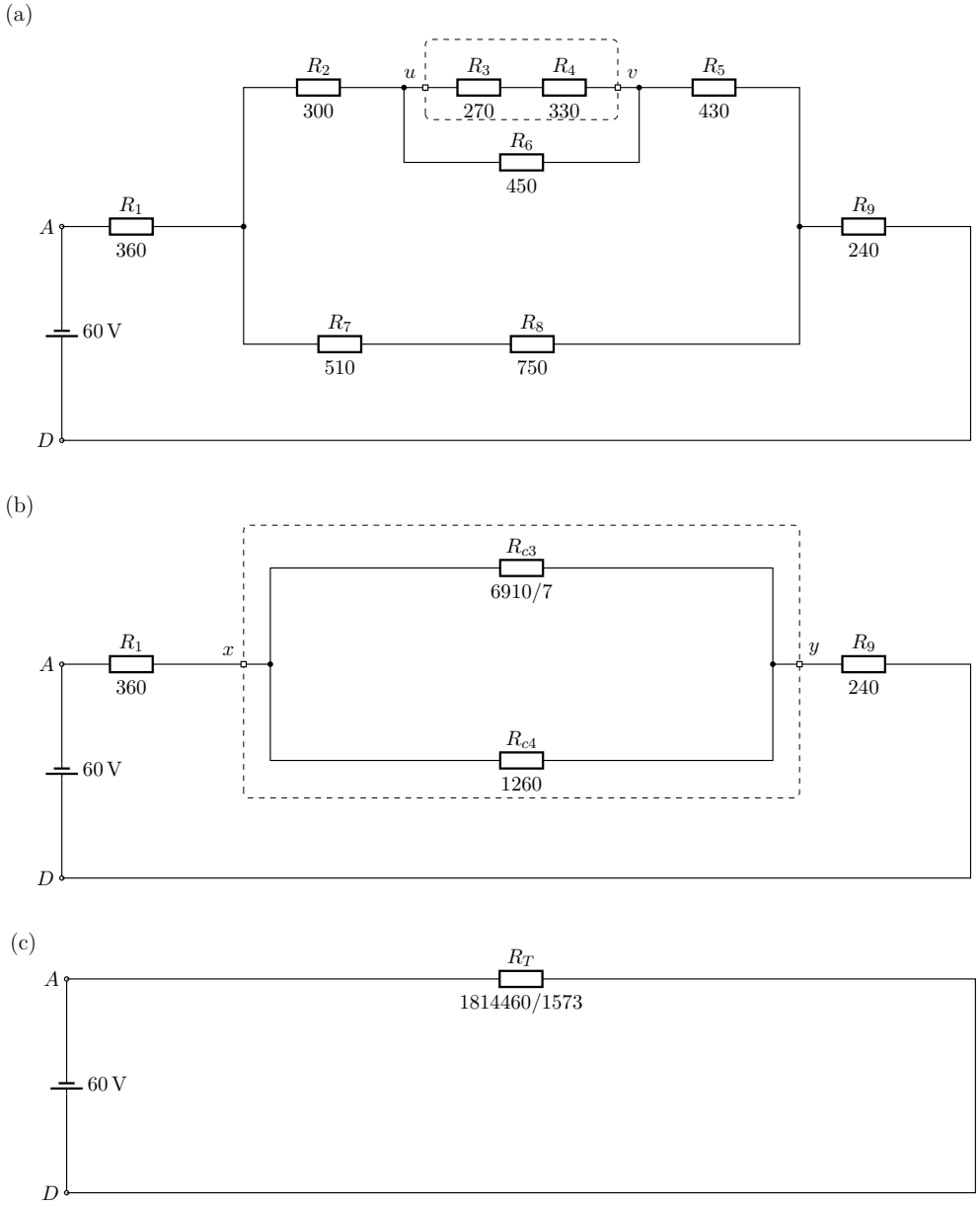


Figure 1: The nine-resistor network of Example 4.4 with a 60 V source across the terminals A and D, resistances in ohms. (a) The original network N_0. The dashed region encloses $\mathsf{S}(\mathsf{R}(R_3), \mathsf{R}(R_4))$, a ready bubble (the leftmost, under the convention of Section 6); it communicates with the rest of the network only through the boundary vertices u and v, which are the same electrical nodes as the junctions it shares with R_6. (b) The network after steps a through d, in which $R_{c3} = R_2 + R_{c2} + R_5 = 6910/7$ and $R_{c4} = R_7 + R_8 = 1260$. The dashed region encloses the parallel pair $R_{c3} \parallel R_{c4}$, the next ready bubble, with boundary vertices x and y. (c) The final equivalent resistor $R_T = 1814460/1573 \approx 1153.503$. Squares mark the points where a wire crosses the boundary of a dashed region, which are the boundary vertices of that region; filled dots are junctions and open circles are the terminals A and D. Each replacement preserves the terminal behavior of the replaced region, hence the source current 52.015 mA, while the reduced representation no longer exposes the currents and voltages internal to the region.

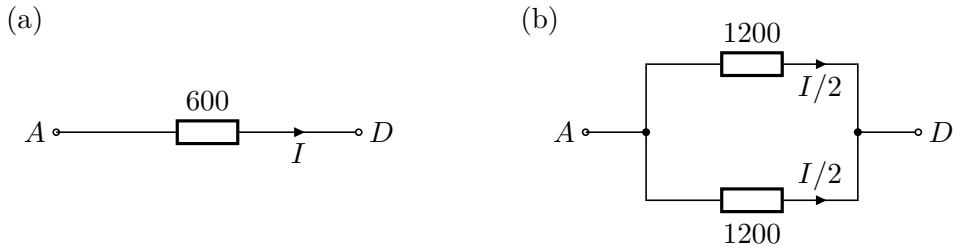


Figure 2: Two networks with the same terminal law, resistances in ohms. (a) A single resistor $\mathsf{R}(600)$. (b) Two resistors of $1200\,\Omega$ in parallel. Under a nonzero terminal current I the resistor in (a) carries I and each resistor in (b) carries $I/2$, so equal boundary behavior leaves the internal current distribution undetermined.

Proposition 4.6 (Recovery from the log)**.** *Let N be a series-parallel network term and suppose the log of a complete reduction of N to a single resistor is retained, each entry recording the replaced subterm, its value, and its position at the time of execution, and suppose the final term is retained. Then reverse replay of the log reconstructs the original decomposition term, and for any terminal voltage V the voltage across and current through every resistor of N is then determined by a single downward pass from the root of the decomposition to the leaves. For a partial log, reverse replay still reconstructs the term, but an upward resistance calculation must precede the downward pass. Conversely, the terminal law alone does not determine them (Proposition 4.5); recovery requires the structure and the values, which the log is one way of supplying.*

Proof. Replaying the entries in reverse order, each replaced subterm is restored at its recorded position, which reconstructs the decomposition. The downward pass is then an induction on the tree. At the root the pair $(V, V/\rho(N))$ is known. If a node is $\mathsf{S}(N_1, \ldots, N_k)$ with current I then the current in each N_i is I and its voltage is $\rho(N_i)I$. If a node is $\mathsf{P}(N_1, \ldots, N_k)$ with voltage V' then the voltage across each N_i is V' and its current is $V'/\rho(N_i)$. The values $\rho(N_i)$ are exactly the values stored in the log. □

Resistor	Resistance (Ω)	Current (mA)	Voltage (V)	Power (W)
R_1	360	52.015	18.726	0.9740
R_2	300	29.166	8.750	0.2552
R_3	270	12.500	3.375	0.0422
R_4	330	12.500	4.125	0.0516
R_5	430	29.166	12.541	0.3658
R_6	450	16.666	7.500	0.1250
R_7	510	22.850	11.653	0.2663
R_8	750	22.850	17.137	0.3916
R_9	240	52.015	12.484	0.6493

Table 1: Internal state of the network of Example 4.4 under the 60 V source, recovered from the log by the downward pass of Proposition 4.6 in exact arithmetic. The reduced network $\mathsf{R}(R_T)$ determines only the total current of 52.015 mA.

Table 1 gives the recovered state. The voltage drops across R_1, the middle block, and R_9 sum to 60 V, and the total dissipation is 3.121 W in both the original and the reduced

network, which agrees with the preservation of the terminal pair. The distribution of that power among the nine original resistors is available only because the log was kept. The textbook computes the first part of this state with the current rounded to 0.052 A and so reports 18.72 V across R_1 and 12.48 V across R_9, which differ in the third decimal place from the exact values above. Rounding is a small instance of the general point of the next section: once finite-precision arithmetic is involved, the order and the intermediate terms of a reduction can leak into the answer, and the history is no longer merely an accompaniment of the value. This observation is the bridge from reduction to history: what the reduced representation no longer exposes can be restored whenever sufficient structural information is available, whether from a retained log or from some other source such as the original circuit diagram, and the question of which source is authoritative is a question about the semantics of the operation and not about its arithmetic.

5 Where the Correspondence Fails

In what follows bare resistance symbols such as R_1 abbreviate resistor terms $\mathsf{R}(R_1)$. It is not claimed that every circuit transformation is a SpherePOP operation, and the claim should not be strengthened to that. The correspondence has limits of more than one kind. Pairs that agree at one level and disagree at another were given after Definition 3.3. A different kind is the circuit that cannot be written as a nested term at all.

Proposition 5.1 (The bridge)**.** *The Wheatstone bridge, with nodes A, B, C, D, edges AB, AC, BD, CD, BC and terminals A, D, is not the graph of any series-parallel network term.*

Proof. Suppose it were. Every series-parallel term other than a single resistor has a root that is either a series or a parallel composition. A root of the form $\mathsf{P}(N_1, \ldots, N_k)$ with $k \geq 2$ would make the graph the union of k two-terminal graphs meeting only at A and D. The bridge has no edge AD, so no component consists of A and D alone, and each component therefore contains an internal vertex. Distinct components share no internal vertex and no edge, so no path joins internal vertices of different components once A and D are deleted, and the deletion would leave at least $k \geq 2$ connected components. Deleting A and D from the bridge leaves the single edge BC, a connected graph with one component. A root of the form $\mathsf{S}(N_1, \ldots, N_k)$ with $k \geq 2$ would produce a cut node other than A and D separating them. Deleting B leaves the path A, C, D, and deleting C leaves the path A, B, D, so there is no such node. Neither form is possible. □

The bridge is the standard minimal obstruction to series-parallel decomposition, and a classical characterization of series-parallel two-terminal graphs by forbidden configurations is due to Duffin [5]. Its precise hypotheses on terminals and subdivisions are not needed here, because the proof above is self-contained for the bridge with its terminals at A and D, and whether a graph is series-parallel depends on its designated terminals. A two-terminal circuit whose graph is the bridge cannot be reduced to a number by nested series and parallel replacements alone. The terminal resistance of such a circuit can still be found by solving the Kirchhoff equations directly, and the bridge circuit is a correct and useful circuit, but its solution is not an instance of popping nested bubbles.

Two refinements show that the failure is informative. In the special case in which the bridge is balanced, meaning $R_1/R_2 = R_3/R_4$ for the resistors on the four outer edges, the

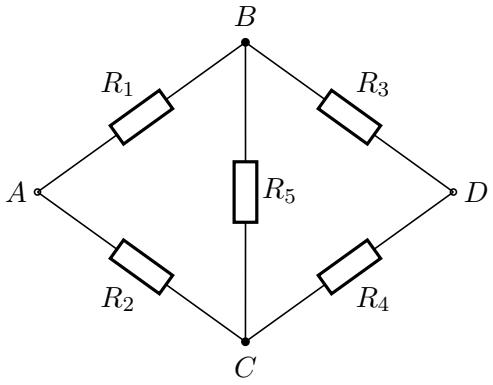


Figure 3: The Wheatstone bridge with supply terminals A and D. The resistor R_1 lies on AB, R_2 on AC, R_3 on BD, R_4 on CD, and R_5 across the middle on BC. Deleting A and D leaves the single edge BC, and no vertex other than A and D separates A from D, so the graph is neither a series nor a parallel composition. In the balanced case $R_1/R_2 = R_3/R_4$ the resistor R_5 carries no current and may be deleted, leaving the chains $A-B-D$ and $A-C-D$ in parallel.

central resistor R_5 carries no current, and it may be deleted, leaving $\mathsf{P}(\mathsf{S}(R_1, R_3), \mathsf{S}(R_2, R_4))$. With $R_1 = 100$, $R_2 = 200$, $R_3 = 300$, $R_4 = 600$ the balance holds and the resistance is $400 \cdot 800/1200 = 800/3 \approx 266.667\ \Omega$. The point is that the deletion is licensed by a condition on the values and not by the shape of the network. A rule of this kind has a precondition that cannot be read off the drawing, and a system that applies it must be able to test the precondition and to decline when it fails. This is the first appearance of refusal as an operation in its own right, and Section 7 returns to it.

The second refinement is the transformation that handles the unbalanced bridge. The star-delta transformation [8], shown in Figure 4, replaces a triangle of three resistors by a star with a new central node, preserving the behavior at the three outer nodes. In the bridge, the triangle on A, B, C with resistors R_1, R_2, R_5 is replaced by a star with resistances $R_a = R_1R_2/\Sigma$, $R_b = R_1R_5/\Sigma$, and $R_c = R_2R_5/\Sigma$ where $\Sigma = R_1 + R_2 + R_5$, joining a new node O to A, B, and C respectively. The resulting network is $\mathsf{S}(\mathsf{R}(R_a), \mathsf{P}(\mathsf{S}(\mathsf{R}(R_b), \mathsf{R}(R_3)), \mathsf{S}(\mathsf{R}(R_c), \mathsf{R}(R_4))))$, which is series-parallel. For $R_1 = 100$, $R_2 = 200$, $R_3 = 300$, $R_4 = 400$, and $R_5 = 150$, the star resistances are $400/9$, $100/3$, and $200/3$ ohms and the terminal resistance is $2150/9 \approx 238.889\ \Omega$, a value that agrees with a direct nodal solution. This transformation is not a replacement of a two-terminal region by a single resistor. Its input is a region with three boundary vertices, its output has a different number of nodes, and the bubble drawn around the triangle has no counterpart as a child of a single parent in a nested term. It therefore shows that the circuit domain supplies a larger class of local replacements than the nested calculus does, and that to include it one needs regions exposing more than two boundary vertices. Whether such regions can be added to SpherePOP without sacrificing the properties established above is a proposed extension and not a result of this essay.

The bridge and the star-delta transformation expose a distinction that is easily lost when nested circles are compared with nested circuits. A nested expression is a tree, and an electrical network is in general a graph. Two computational models are available, and they are different. In the first, the state is the network graph itself, connectivity is

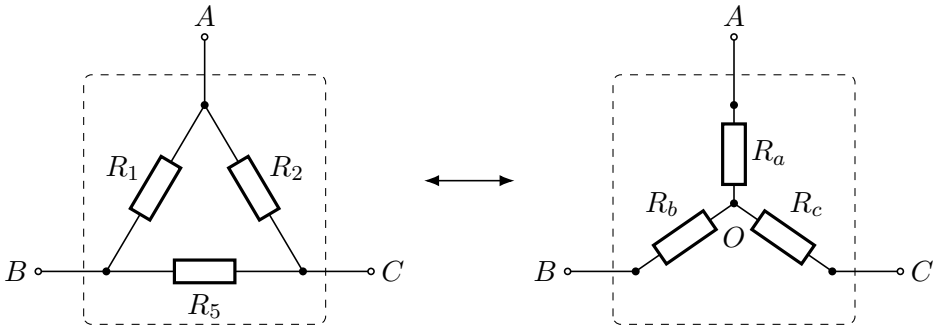


Figure 4: Replacement of the triangle of R_1, R_2, R_5 by a star with a new interior vertex O. The boundary vertices A, B, C are the same on both sides, and the complete three-terminal voltage-current relation is preserved, which is the property that licenses replacement in an arbitrary surrounding circuit, with $R_a = R_1R_2/\Sigma$, $R_b = R_1R_5/\Sigma$, $R_c = R_2R_5/\Sigma$, and $\Sigma = R_1 + R_2 + R_5$. The region has three boundary vertices, and its interior connectivity changes, so the replacement is not a replacement of a two-terminal region by a single resistor.

represented directly, and the operations are local replacements of regions of the graph by equivalent regions; a three-terminal star-delta step is natural in this model and no tree structure is needed. In the second, the state is a term, the graph is the semantics of the term, and a term is a record of a decomposition imposed on connectivity, so that a term exists only for those graphs that admit the decomposition. Series-parallel graphs admit decomposition terms [4], and Theorem 3.6 identifies the transitions of a given term with those of the network it denotes. Representation does not identify the two operational models, however: whether the graph-level and term-level transition systems correspond depends on the permitted reassociations and on the granularity of the reductions, as the example after Theorem 3.6 shows. Beyond series-parallel networks they diverge: the bridge has a graph and no term, and the star-delta step has a graph-level description and no term-level counterpart without a richer syntax. Whether SpherePOP is to be understood as the first model, with connectivity represented directly, or the second, with the nested structure as a history of transformations applied to connectivity, is a design decision that the circuit example does not settle but does bring into the open, and the choice determines which of the proposed extensions are extensions of the calculus and which are changes of model.

6 From Reduction to Executable History

A set of valid replacements does not constitute an execution. To execute, one needs a strategy that selects among the replacements that are available at each moment, and the choice made determines the history. The preceding sections have shown that the final value is independent of the choice. The histories are not.

Consider a flat series chain of k resistors whose values are generic, in the sense that all subset sums are distinct, and consider a separately defined *binary aggregation rule*, which replaces any two children a, b of a series bubble that has at least three children by the single numeral $a + b$, leaving the other children in place, and a final ordinary `POP` applies when two children remain. For arbitrary pairs the children are treated as an unordered collection, so that no placement convention for the sum is needed, and for adjacent pairs the sum occupies the place of the two. The rule combines any two children, which for series composition

presupposes commutativity and is the electrically equivalent rearrangement of the remark in Section 3 and not a replacement of a region of the original chain; if only adjacent children may be combined, the number of complete sequences is $(k-1)!$ in place of the count below. This rule is distinct from the all-children `POP` of Section 2, under which a flat series bubble with numeral children reduces in one step, and it describes a finer-grained calculus. At the stage when m children remain, there are $\binom{m}{2}$ pairs that could be combined, and each choice gives a different term. The number of maximal reduction sequences is therefore

$$\prod_{m=2}^{k} \binom{m}{2} \;=\; \frac{k!\,(k-1)!}{2^{k-1}},$$

which is $1, 3, 18, 180, 2700$ for $k = 2, \ldots, 6$. All of these sequences end at the same numeral. The count is of complete sequences of choices and not of globally distinct intermediate terms, since different histories can converge at an intermediate state, and with generic values the sequences are pairwise distinct as sequences. A history in the sense of a recorded log is therefore not determined by the problem, and the number of possible histories grows much faster than the number of resistors.

Three objects should be kept apart in discussing such counts. For the remainder of this section executions use only the original all-children `POP` relation, and the binary aggregation rule is not considered. An *execution* is a particular run $t_0 \longrightarrow t_1 \longrightarrow \cdots \longrightarrow t_m$. Two steps of an execution are *independent* when the bubbles they pop occupy disjoint positions, neither containing the other, and exchanging two adjacent independent steps produces another execution with the same final term. The *dependency order* of an execution is the least partial order on its steps in which a step precedes a later step whenever the bubble popped by the later step contains the numeral produced by the earlier one. Executions related by exchanges of adjacent independent steps are identified in a *trace* [11], and a trace is determined by its dependency order, its members being the linear extensions of that order, provided that the events carry labels, which within a fixed initial term are the position of the popped bubble and the subterm it replaces, while comparing traces across different initial terms also requires the full initial terms or an equivalence between them: each step is labeled by the persistent identity of the original bubble occurrence it pops and by its operator, since an unlabeled partial order could not distinguish computations on different values or operators. A *serialized record* is a single linear listing of steps, which is a member of a trace chosen for storage or display. A strategy such as leftmost innermost selects one serialized record from each trace it produces, but the record is not the trace: two strategies can yield different records of the same trace.

The textbook network makes the distinction concrete. With n-ary `POP`, its reduction has six steps, named $a = R_{c1}$, $b = R_{c2}$, $c = R_{c3}$, $d = R_{c4}$, $e = R_{c5}$, and $f = R_T$. The dependency order is generated by $a < b < c < e < f$ and $d < e$. The step d pops $(+\ 510\ 750)$, which is disjoint from the nested bubbles popped by a, b, and c, so d is independent of each of them. The linear extensions are the four sequences obtained by inserting d at one of four positions among a, b, c, and each is reached from the others by exchanges of adjacent independent steps, which proves that the four executions form a single trace. The order printed in the textbook, with d after c, is the serialized record that results from popping the leftmost ready bubble at each stage, so the printed solution is the record selected by the innermost-first strategy with the leftmost tie-breaking convention adopted here. The

innermost-first instruction fixes no tie-breaker among ready bubbles, and the textbook sequence agrees with the leftmost convention adopted here. The printed order is one of four equally valid serializations of one trace.

Proposition 6.1 (Executions of a fixed term)**.** *Let t be a bubble term. The bubble occurrences of t form a finite partial order in which every descendant bubble precedes each of its ancestors. The complete executions of t under the all-children* `POP` *are exactly the linear extensions of this order, and any two complete executions of t are related by exchanges of adjacent independent steps, so all complete executions of t form a single trace.*

Proof. A bubble occurrence is ready exactly when all bubble occurrences below it have been popped, so a sequence that pops each occurrence once and pops every descendant before its ancestors is executable, and every complete execution has this form. Two linear extensions of a finite partial order are connected by a sequence of exchanges of adjacent incomparable elements, by the standard argument that a minimal element of one extension can be moved to the front of the other by such exchanges. Adjacent incomparable occurrences are not in the ancestor relation, so the bubbles they pop occupy disjoint positions and the steps are independent. □

The proposition is the general form of the textbook count, within the restricted calculus defined here. The distinction has consequences for what a system should store. A system that stores serialized records distinguishes four executions of the network that are mathematically indistinguishable in their dependency structure. A system that stores only the dependency order identifies them directly. A serialized record selects one ordering, whereas a dependency representation expresses the permitted reorderings directly. Neither is inherently unable to represent what the other conveys under the assumptions of this essay: a record that retains the replaced subterms and positions reconstructs the initial term, and together with the `POP` rules that term determines the other permissible orders, so alternatives are derivable even when they are not enumerated, and two records can be compared modulo independent exchanges by any system that implements that equivalence.

The relation of this structure to the confluence theory of rewriting is worth stating without overstating it. For the calculus of this essay confluence in the usual sense is easy: the system terminates by Lemma 2.3 and every maximal sequence reaches the same numeral, so the normal form is unique [1, 2]. What confluence does not provide is any claim about the intermediate terms, and the quantity that distinguishes one history from another is exactly the quantity that the confluence theorem ignores. The strategy-independence of the value and the strategy-dependence of the history are two separate facts, and a claim at the third level is a claim about the second.

7 Pop, Merge, Collapse, and Refuse

If `POP` were the only operation, SpherePOP would be a graphical evaluator for arithmetic expressions in the style of the classical machines for mechanical expression evaluation [10], and the circuit example would exhibit nothing that the circles-of-evaluation notation did not already contain. The circuit example is informative because it forces the questions that a four-operation vocabulary is designed to answer. The definitions below are those used in this essay, offered as a reading of the calculus under which the circuit analogy can be stated

precisely. They should be checked against the operational definitions of the calculus itself, and where they differ the circuit analysis shows what each operation would have to do.

`MERGE` is the structural operation that joins two regions into one without evaluating either. In the circuit domain it is the identification of a series bubble nested inside another series bubble with a single flat series bubble, $\mathsf{S}(a, \mathsf{S}(b, c)) = \mathsf{S}(a, b, c)$, which is an application of associativity and does not change any value. A reduction that proceeds by popping pairs presupposes a decision about whether the nested form or the flat form is the working representation, and `MERGE` makes that decision a visible step. The count of $k!(k-1)!/2^{k-1}$ in Section 6 depends on the flat form; for the fully nested binary form the corresponding count is smaller, and the difference is the effect of the merging convention on the space of histories.

The four operations are best stated as transitions on configurations, so that their differences lie in their preconditions and effects and not in whether some external mechanism happens to keep a record. A *configuration* is a pair (t, h) of a bubble term and a finite sequence h of log entries, the log being part of the state of the calculus. A position π is a path from the root of the term tree, and every entry is interpreted relative to the term at the moment it was appended. The rules below are stated with $C[\,\cdot\,]$ a context in which the hole is at position π. `POP`, `COLLAPSE`, and `REFUSE` replace or leave the subterm at π and do not change the path of any subterm outside it, so entries that concern positions outside π stay valid after the step. `MERGE` does change the paths of the siblings that follow the merged bubble, and its entry therefore records the position π of the parent bubble, the index j at which the flattened bubble began, and its arity m, so that the children $j, \ldots, j+m-1$ of the result can be regrouped and the paths of the following siblings restored. Nested merges introduce no further obstruction. With the parent position, index, and arity recorded and undo following the stack discipline described next, undoing the later operations restores exactly the configuration in which the inverse of an earlier merge applies, and the claim follows by induction on the number of operations undone. Inversion is given by explicit undo transitions, `UNPOP` and `UNMERGE`, each defined only on a configuration whose most recent log entry is the corresponding `POP` or `MERGE` entry. `UNPOP` takes $(C[n], h \cdot \langle \pi, b \rangle)$ to $(C[b], h)$, and `UNMERGE` uses the recorded index and arity to restore the grouping and the paths of the siblings. A third undo transition, `UNREFUSE`, removes the most recent refusal entry and leaves the term unchanged. A checkpoint entry delimits undo: no undo transition applies below the most recent checkpoint. Inversion follows a stack discipline within the log above the checkpoint: only the most recent entry can be undone directly, and earlier entries are undone by undoing the later ones first, or equivalently by replaying the log from the checkpointed configuration. In particular a `POP` entry that precedes a `MERGE` entry in the log can be undone only after that merge has been undone, which the stack discipline enforces, so a position recorded before a merge is never interpreted against the merged term. Invertibility here is formal, relative to the configuration and its log. It is distinct from replay-based reconstruction of a state from a stored initial term and the log, and from physical reversibility of the process that carried out the steps, which is not asserted.

`POP` takes $(C[b], h)$, where $b = (\oplus\ n_1 \cdots n_k)$ is ready, to $(C[n],\, h \cdot \langle \pi, b \rangle)$ with $n = \oplus(n_1, \ldots, n_k)$. Its precondition is readiness and its effect is to replace the bubble by its value and to append an entry that records the replaced bubble. It is invertible on configurations: from $(C[n], h \cdot \langle \pi, b \rangle)$ the configuration $(C[b], h)$ is recovered, which is the content of Proposition 4.6 stated for the log of the calculus. `COLLAPSE` takes $(C[s], h)$, where

s is any subterm at π, to $(C[[\![s]\!]], \langle \text{checkpoint}, C[[\![s]\!]]\rangle)$. Its precondition is that s be a closed region, and for bubble terms every subterm is. Its effect is to evaluate the entire region in a single step and to replace the whole log by a single checkpoint entry that holds the resulting term and nothing earlier. The erasure is global, which avoids the difficulty that log paths are interpreted relative to different historical states and that an entry outside a region can retain the region's former structure. A selective collapse that erases only the history associated with one region while keeping every remaining entry valid would need stable region identities and explicit dependencies among entries, and it is left open. The difference from `POP` is therefore a difference in the transition rule: `POP` appends an entry that determines the replaced bubble, and `COLLAPSE` replaces the log by a checkpoint that does not.

Proposition 7.1 (Collapse is irreversible)**.** *For a fixed collapse position π, the transition `COLLAPSE` at π is noninjective: there are configurations $(C[s], h)$ and $(C[s'], h')$ with $s \neq s'$, whatever the logs h and h', that it takes to the same configuration. This shows that `COLLAPSE` is a noninjective checkpointing operation, and it does not show that selective erasure of the history of a single region has been given a correct definition. Consequently `COLLAPSE` at π has no left inverse recovering every predecessor, whereas `POP` has one.*

Proof. The irreversibility is information-theoretic relative to the configuration and not a physical statement. Take $s = (+\ 270\ 330)$ and $s' = (+\ 300\ 300)$. Both have value 600, so collapsing either at position π gives $(C[600], \langle \text{checkpoint}, C[600]\rangle)$ whatever the logs h and h' were. By contrast the `POP` entries $\langle \pi, s\rangle$ and $\langle \pi, s'\rangle$ differ, so the configurations after popping differ. □

The equivalent resistor R(600) in place of the two original resistors is the circuit counterpart of a collapsed region. It satisfies Proposition 4.2, so the rest of the network is unaffected, and by Proposition 4.5 it has abandoned information that the result alone does not determine. The difference between popping and collapsing is a difference in commitment that is invisible to the arithmetic and is fixed by the rules. Recovery is possible after a `POP` because its log entry retains the replaced bubble, and `COLLAPSE` is irreversible because its transition replaces the whole log by a checkpoint that discards that entry. A sequence of pops followed by erasure of the log by the environment is not a collapse, because that erasure is not a transition of the calculus. A collapse performs the erasure as its own transition, and the point of making the log part of the configuration is that the calculus can state which of its own transitions are reversible.

The transition system is labeled by attempted operations $\alpha = \langle \text{op}, \pi\rangle$ with $\text{op} \in \{\texttt{POP}, \texttt{MERGE}, \texttt{COLLAPSE}\}$, where an attempt of `MERGE` also names the index j of the child of the bubble at π that is to be flattened into it, so that the attempt is unambiguous when several children carry the same operator. For an attempt at a position that exists, refusal takes the form displayed below. For an attempt at a position π that does not exist, it takes the form $(t, h) \xrightarrow{\alpha} (t,\ h \cdot \langle \alpha, \text{refusal}, \text{no such position}\rangle)$, since the context notation $C[s]$ applies only to positions that exist. A configuration answers an attempt either by performing the operation, when its precondition holds, or by `REFUSE`: $(C[s], h) \xrightarrow{\alpha} (C[s],\ h \cdot \langle \alpha, \text{refusal}, \rho\rangle)$ when the precondition of α fails, where ρ names the failed precondition. The term is unchanged and the log records the attempted label and why it failed, which is what distinguishes a refusal from an arbitrary append. The precondition of Proposition 4.2, that the region be two-terminal-isolated, is the clearest case: a bubble drawn around a

subnetwork with a third boundary vertex is not a candidate for replacement, and a system that attempted the replacement would produce a wrong answer without signaling any error. The balanced-bridge deletion is another, because the condition $R_1/R_2 = R_3/R_4$ must hold for the rule to apply. In a language of expressions refusal arises for attempts at positions that do not exist, for a `POP` at a bubble that is not ready, and for a `MERGE` whose selected child carries a different operator from its parent, while every ready bubble admits a `POP`, so that a strategy that selects only ready bubbles never needs a readiness refusal. The circuit-specific refusals, two-terminal isolation and the balance condition, require a graph-state language that the configuration syntax of this essay has not supplied, and they are described here at the level of the circuit and not as rules of the configuration system. Interface specifications of this kind illustrate the need for stated preconditions, as in the calling conditions of an assembler's parser, whose accessor for the destination field is specified for computation commands only and whose advance routine presupposes that another command exists [12]. Such specifications do not themselves define a logged refusal, so they motivate the form of the operation without establishing the semantics proposed here. In a language where reduction rules carry preconditions on regions and on values, refusal is a first-class outcome of attempting a step, and treating it as a failure to be hidden is the source of the silent errors that arise in applying a series-parallel rule to a circuit that is not series-parallel.

`MERGE` also acts on configurations: it takes $(C[(\oplus\ s_1 \cdots (\oplus\ u_1 \cdots u_m) \cdots s_k)], h)$ to the flattened bubble with an entry recording the grouping, for any associative $\oplus$. These four operations differ in kind and not merely in name. `MERGE` preserves values and, with its entry, structure. `POP` changes the structure, preserves the value, and is invertible. `COLLAPSE` changes the structure, preserves the value, and is not invertible. `REFUSE` changes only the log. The circuit example motivates the requirements that these operations are proposed to meet, namely preserving structure, checking preconditions, and distinguishing recoverable from lossy transformations, which are the quantities that Section 4 shows a reduction to be sensitive to. It does not establish that the requirements must take precisely the proposed forms of `MERGE`, `COLLAPSE`, and `REFUSE`. The plain circles-of-evaluation notation has no place for them. A parenthesized expression that is evaluated to a number has no question to ask about what was lost.

8 Further Work: Integration with the SpherePOP Kernel

The restricted evaluator developed here supplies a concrete domain for investigating integration with the executable SpherePOP kernel [15]. The kernel separates immutable terms, rule-governed reduction, execution records, scheduling, and host-authorized effects. Its present reduction language consists of the combinators I, B, C, K, W, and S, permission-gated and applied within scopes, together with two boundary rules, absorption and scope, and its numerals are inert atoms that only a host reads as integers. It does not contain the rational series and parallel operations used in this essay. Integration therefore requires an explicit representation and an account of which transitions belong to the reducer and which belong to a circuit host. The correspondence proved here does not by itself establish that integration.

Two implementation routes are available. The first represents positive rational numbers and the series and parallel operations as programs in the existing combinatory language. This preserves the kernel's rule table, but a single circuit-level `POP` would generally require

several kernel reductions, so the appropriate correctness statement would relate one circuit step to a finite sequence of kernel steps. The second route introduces a separately specified arithmetic extension with exact rational values and explicit series and parallel contraction rules. This could preserve the step granularity of the present evaluator, but it would require corresponding changes to the term representation, the rule specification, the serialization, and the verifiers. Neither route should identify a circuit step with a kernel step merely because their final values agree.

Let E encode bubble terms as kernel states and let D decode completed arithmetic representations. A basic obligation is semantic preservation,

$$D(\mathrm{Eval}_K(E(t))) = [\![t]\!],$$

whenever the encoded evaluation completes under the stipulated configuration and strategy. For an encoding into the existing language, an additional operational obligation is

$$t \longrightarrow t' \quad \Longrightarrow \quad E(t) \longrightarrow_K^+ E(t'),$$

or a corresponding relation between representations when the encoding introduces administrative structure. The converse must distinguish administrative reductions from completed circuit operations. Termination of the source calculus does not establish termination of the encoding without a further argument, and fuel exhaustion must remain distinct from a completed resistance calculation.

A circuit host would retain connectivity, designated terminals, resistor identities, and exact resistance values. A proposed replacement would identify the selected component and the replacement network, and the host would check the conditions that make the operation admissible, which for the present domain include two-terminal isolation and the series or parallel composition pattern. The kernel's permissions govern contraction of combinators within scopes. They do not themselves establish electrical isolation or equality of terminal laws. A verified kernel execution and an admissible circuit replacement are therefore separate claims, joined by the correctness of the encoding and the host's validation procedure.

The kernel's separation between reduction and host effects offers a suitable architectural starting point. A program could produce an inert request to replace an identified circuit region, after which a circuit-specific schema and validator would determine whether the host commits or refuses the request. Such a validator would need replayable evidence of the selected region, its boundary, and the calculation justifying equivalence. The kernel's own documentation draws the corresponding caution: a verified effects log shows that dispositions were recorded in a chain and that each request is consistent with the schema, but it does not show that the authorizer decided correctly unless its policy and inputs are replayed. In the same way, checking that a disposition was recorded would not establish that the electrical decision was correct unless the relevant circuit state and validation conditions were also checked.

The history-sensitive operations require a separate integration argument. The kernel specification lists the disposition calculus among the elements not implemented in its first milestone, and states the intention that it live in the evidence layer and not in the reducer. In particular, the global log-erasing `COLLAPSE` of Section 7 should not be identified with an append-only audit event, which is the form the kernel's disposition log takes. An integration could distinguish a working configuration, whose reconstruction information

may be discarded, from an audit record that documents the decision to discard it. Any irreversibility claim would then specify which state and which retained records are available to an observer. Retaining enough information elsewhere to reconstruct the predecessor would defeat a claim of global information loss, although the working configuration alone could remain noninvertible.

An initial implementation should use the nine-resistor example as a conformance case. It should reproduce the exact terminal resistance, preserve the correspondence between source regions and execution records, reconstruct the original resistor states, and reject altered records or invalid replacement requests. Independent checking should cover the arithmetic and electrical preconditions as well as the kernel reductions. This would turn the present interpretation into a testable implementation while preserving the distinction between computing a value, verifying its derivation, and authorizing a change to a circuit representation.

9 What Has and Has Not Been Established

The claim that series-parallel reduction is SpherePOP in action can now be stated as a result with a stated domain. Within the domain of series-parallel resistor networks, the nested bubble calculus with the `POP` rule is a sound model of terminal-resistance reduction, relative to a chosen decomposition term and the stipulated all-children rules. This means precisely that the translation τ is a step-for-step correspondence between replacement in networks and `POP` in terms (Theorem 3.6), that each step is a valid replacement under the condition of two-terminal isolation (Propositions 4.2 and 4.3), that the final value is independent of the strategy (Lemma 2.3), and that the internal state is recoverable when sufficient structural information, such as the log, is retained but not from the terminal law alone (Propositions 4.5 and 4.6). Outside this domain the correspondence fails or must be supplemented: the bridge is not a nested term (Proposition 5.1), the balanced-bridge deletion requires a value-dependent precondition, and the star-delta transformation requires regions with more than two boundary vertices. The definitions of `POP`, `MERGE`, `COLLAPSE`, and `REFUSE` as transitions on configurations in Section 7, and the irreversibility of `COLLAPSE` (Proposition 7.1), are a proposed reading and are to be checked against the formal definitions of the calculus.

Three extensions are proposed and not proved. The first is a multi-terminal bubble calculus capable of expressing the star-delta transformation and its inverse. The second is the extension of the observation set from the terminal law to dependent sources and nonlinear elements, for which the replacement property must be re-established with a different notion of equivalence, since the argument of Proposition 4.2 used linearity only to describe $\mathcal{B}(M)$ as a line, and the elimination argument itself may survive. The third is the extension of Proposition 6.1 from the all-children `POP` to calculi that include the binary aggregation rule or the operations of Section 7, for which identities and dependencies must be defined for the aggregate components that aggregation produces. None of these is needed for the result to hold in the domain in which it has been checked.

The three relations of Definition 3.3 are the organizing principle of the essay and not only a device of Section 3. Shape resemblance is what a diagram can suggest, a reduction image is what a proof of validity can establish, and step bisimilarity is one correctness relation for an implementation, the position-preserving correspondence of Theorem 3.6 being a stronger

one, and the failures of the analogy are in each case the failure of one relation while another holds. What has been proved is a sound resistor interpretation of a restricted bubble-rewriting calculus, with exact correspondence for chosen decomposition terms, together with a set of precisely stated limits. The proposed history-sensitive operations are a separate investigation that this interpretation motivates, and the relation of the restricted evaluator to an executable kernel is the subject of Section 8. That the restricted evaluator faithfully captures the mature SpherePOP calculus is an open obligation, and the four-operation semantics are a proposal toward discharging it.

A question remains that the circuit example raises without answering. Two configurations may be observationally equivalent, in that their terms have the same value and the same boundary behavior, without being equivalent as histories. The distinction can be seen directly in the two `POP` executions of (+ 270 330) and (+ 300 300), which both produce the term 600 while retaining different entries. Historical equivalence may be taken to mean literal equality of logs or equality modulo independent exchanges, and the question of when it coincides with observational equivalence has a different answer under each. Proposition 4.5 establishes only that internal electrical structure is underdetermined by terminal behavior, and Proposition 7.1 shows that a calculus can erase the difference between histories by a rule. Under what conditions the two equivalences coincide, and whether the difference between them should be a mathematical object of the calculus and not metadata attached to a finished computation, is the question to which a fuller development would turn.

The general lesson concerns how a proposed correspondence between a notation and a calculus should be assessed. A resemblance of shape does not contain the theory it suggests, and it is not empty merely because it lacks that theory. It identifies a relationship from which a theory can be developed, and the test of the identification is whether the development can be carried out, whether the conditions under which it holds can be stated, and whether the points at which it fails are informative. In this case the development can be carried out for a well-defined class of circuits, the conditions are two-terminal isolation and positivity of the resistances, and the failures, the bridge and the three-terminal region, mark the places where the nested calculus must either be extended or declined. The circuit analogy suggests a reading more precise than the resemblance of nested circles to nested arithmetic. On that reading SpherePOP is a calculus of admissible local replacement, in which equivalence is defined at the boundary of a region and the log records which distinctions the replacement eliminated.

References

[1] A. Church and J. B. Rosser, Some Properties of Conversion, *Transactions of the American Mathematical Society* 39 (1936), 472–482.

[2] F. Baader and T. Nipkow, *Term Rewriting and All That*, Cambridge University Press, 1998.

[3] R. L. Boylestad, D. Edgar, J. Jenness, and J. Tompkin, *Boylestad's Circuit Analysis*, 2nd Canadian ed., Pearson Education Canada, 2000, Chapter 7, Series-Parallel Networks, pp. 144–145.

[4] J. Valdes, R. E. Tarjan, and E. L. Lawler, The Recognition of Series Parallel Digraphs, *SIAM Journal on Computing* 11 (1982), 298–313.

[5] R. J. Duffin, Topology of Series-Parallel Networks, *Journal of Mathematical Analysis and Applications* 10 (1965), 303–318.

[6] M. Felleisen, R. B. Findler, M. Flatt, and S. Krishnamurthi, *How to Design Programs*, 2nd ed., MIT Press, 2018.

[7] S. L. Herman, *Delmar's Standard Textbook of Electricity*, 5th ed., Delmar, Cengage Learning, Clifton Park, NY, 2011, Section II, Unit 8, Combination Circuits, Figures 8–9 through 8–15, p. 213.

[8] A. E. Kennelly, Equivalence of Triangles and Three-Pointed Stars in Conducting Networks, *Electrical World and Engineer* 34 (1899), 413–414.

[9] G. Kirchhoff, Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird, *Annalen der Physik und Chemie* 72 (1847), 497–508, doi:10.1002/andp.18471481202.

[10] P. J. Landin, The Mechanical Evaluation of Expressions, *The Computer Journal* 6 (1964), 308–320.

[11] A. Mazurkiewicz, Trace Theory, in *Petri Nets: Applications and Relationships to Other Models of Concurrency*, Lecture Notes in Computer Science 255, Springer, 1987, 279–324.

[12] N. Nisan and S. Schocken, *The Elements of Computing Systems: Building a Modern Computer from First Principles*, MIT Press, Cambridge, MA, 2005, Chapter 6, Assembler, pp. 112–113.

[13] J. H. Morris, *Lambda-Calculus Models of Programming Languages*, Ph.D. thesis, Massachusetts Institute of Technology, 1968, also issued as Project MAC Technical Report MAC-TR-57.

[14] G. D. Plotkin, A Structural Approach to Operational Semantics, *Journal of Logic and Algebraic Programming* 60–61 (2004), 17–139.

[15] Flyxion, *SpherePOP Kernel*, software repository, specification SPEC.md at commit 511f63e, `https://github.com/standardgalactic/spherepop-kernel/blob/511f63e7b8e68367da633dff5790b8a33720519f/SPEC.md`.

[16] E. Schanzer, K. Fisler, S. Krishnamurthi, and M. Felleisen, Transferring Skills at Solving Word Problems from Computing to Algebra Through Bootstrap, in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 2015, 616–621.