Scroll nets

Pablo Donato
Charles University, Czechia

Abstract

We introduce a new formalism for representing proofs in propositional logic called "scroll nets". Its fundamental construct is the scroll, a topological notation for implication proposed by C. S. Peirce at the end of the 19th century as the basis for his diagrammatic system of existential graphs (EGs). Scroll nets are derived from EGs by following the Curry-Howard methodology of internalizing inference rules inside judgments, just as terms in type theory internalize natural deduction rules. We focus on the intuitionistic implicative fragment of EGs, starting from a natural diagrammatic representation of scroll nets, and then distilling their combinatorial essence into a purely graph-theoretic definition. We also identify a notion of detour, that we use to sketch a detour-elimination procedure akin to cut-elimination. We illustrate how to simulate normalization in the simply typed λ-calculus, demonstrating both the logical and computational expressivity of our framework.

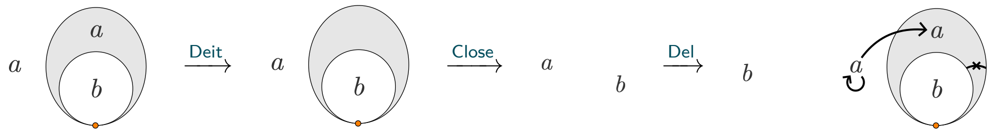
2012 ACM Subject Classification Theory of computation → Proof theory; Theory of computation → Type theory; Theory of computation → Equational logic and rewriting; Theory of computation → Interactive computation

Keywords and phrases deep inference, graphical calculi, existential graphs, Curry-Howard correspondence, simple type theory, cut-elimination

1 Introduction

Graphical proof theory Traditionally, proof theory has been established as a branch of *symbolic* logic: it embodies *par excellence* the Hilbertian ideal according to which any mathematical concept can be expressed as a finite sequence of symbols, with reasoning reduced to the sequential manipulation of these symbols. However, several recent developments tend to demonstrate that a finer analysis of the structure of formal proofs is possible with the help of *graphical* or diagrammatic representations: popular examples include Girard's proof nets in linear logic [3], string diagrams in categorical logic [12], and Hughes' combinatorial proofs [9]. The idea is to abstract from the arbitrary sequentiality suggested by a linguistic usage of symbols, by immersing statements and proofs in space to better understand their geometry. One aim is to find a solution to Hilbert's 24th problem by devising a good notion of *equality* among proofs [26]. However, current research focuses almost exclusively on the structure of *completed* proofs, forgetting the sequential inference process that enabled their construction. This prevents in particular the application of these new methodologies to the design of *interactive theorem provers* (ITPs), which are fundamentally based on the incremental construction of *partial* proofs.

Existential graphs To tackle this limitation, we investigate a graphical system mostly forgotten by contemporary proof theorists, probably because it predates the existence of proof theory itself: C. S. Peirce's *existential graphs* [21] ("*EGs*" hereafter). It is based on a purely diagrammatic and topological representation of logical constants, inspired by a dialogical understanding of reasoning a century before the advent of game semantics [20]. Proving is then modelled as a dynamic process of constructing valid statements through rewriting of diagrams, using six elementary inference rules that perform insertions and deletions of graphs at a single specified location. Figure 1a illustrates how to derive *modus ponens* in EGs, starting from a graph of the statement $a \wedge (a \Rightarrow b)$ and reducing it to b. We refer the reader to [2, Sections 2–4] for a more detailed overview of the history of EGs.



(a) Dynamic proof trace **(b)** Static proof object

Figure 1 Modus ponens in scroll nets

	Proof trace	**Proof object**
Hilbert calculi	Derivation *(sequence)*	Derivation *(sequence)*
Gentzen calculi	Derivation *(tree)*	Derivation *(tree)*
Rocq/Lean	Proof script (= tactic *tree*)	Proof term (= natural deduction *tree*)
Proof nets	—	Formula *tree* + Axiom/Cut *permutation*
String diagrams	Equational rewriting (*sequence*)	—
Existential graphs	Illative transformations (*sequence*)	—
Scroll nets	Illative transformations (*sequence*)	EG *DAG* + Argumentation *forest*

Table 1 Comparing representations of proofs in various formalisms

What is a proof? In order to contemplate the structure of (partial) proofs, one needs a way to represent them *statically* as bona fide *proof objects*. From our perspective, formalisms like Hilbert-style and Gentzen-style calculi conflate the final proof object with what we call the *proof trace* (usually called "derivation"), which is the history of inference steps that were undertaken to construct *dynamically* the proof object. On the other hand, graphical formalisms such as proof nets and combinatorial proofs forego the notion of inference rule altogether, and with it the whole proof trace. Although they are also graphical in nature, EGs somehow lay on the opposite end of the spectrum, together with string diagrams: they lack explicit syntax to capture the structure of proofs, leaving it implicit and mangled in the sequence of rewritings that represents the proof trace. In contrast, state-of-the-art ITPs such as Rocq [27] and Lean [15] do have distinct notions of proof trace and proof object: they are called respectively *proof script* and *proof term*. The full situation is summarized in Table 1.

Scroll nets The aim of the current paper is to introduce a language of *proofs* on top of the existing language of statements in EGs, in order to represent proof objects in addition to proof traces. A notable feature of EGs is that they identify the language of statements with the language of *judgments*: scribing some graph on the *sheet of assertion* is, by definition, the same as asserting its truth[1]. Following the Curry-Howard methodology, our goal is then to internalize inference rules — Peirce called them *illative transformations* — inside the syntax of statements, just like terms in type theory can be seen as a way to internalize and record derivation trees (in natural deduction) inside judgments.

The syntax we have arrived at is based on the simple observation that all illative transformations can be expressed in terms of pure *locations*: they consist in either *inserting/deleting* an arbitrary graph at a given location, *duplicating* an arbitrary graph from a source location, or *deduplicating* an arbitrary graph from a target location. This gives rise to a directed

[1] This identification is also found in the original notation of Gentzen for natural deduction, where nodes in derivation trees are formulas rather than sequents [4].

forest whose nodes are the "subgraphs" of some EG and whose edges encode exactly illative transformations. This data structure is surprisingly close to the *proof nets* of linear logic, and because it is based on the fundamental construct of the *scroll* coming from intuitionistic EGs, we call it ***scroll net***. Figure 1b shows an example of scroll net, which is exactly the proof object built by the derivation of Figure 1a.

Outline The article is organized as follows: in Section 2 we recall the diagrammatic syntax of the implication-conjunction fragment of intuitionistic EGs, based on the fundamental icon of the scroll. In Section 3 we explain how to internalize illative transformations inside EGs by using a diagrammatic arrow notation. In Section 4 we give a formal combinatorial definition of scroll nets, based on a DAG generalization of EGs combined with a forest of illative transformations. In Section 5 we give a sequential correctness criterion for scroll nets following the dynamic understanding of illative transformations, that we use to define horizontal and vertical composition operations. In Section 6 we identify four kinds of detours that arise when a node is both introduced and eliminated, and sketch informally a detour elimination procedure. We also give a direct translation of simply typed λ-calculus into scroll nets, illustrating how to simulate β-reduction. We conclude in Section 7 with a discussion of related works and future directions to improve the metatheory of our framework, extend it to richer logics, and use it in interactive theorem proving applications.

2 Implicative existential graphs

Sheet of Assertion The most fundamental concept of EGs is the *sheet of assertion*, denoted by SA thereafter. It is the space where statements are scribed by the reasoner, typically a sheet of paper, a blackboard, or a computer display. This last analogy suggests an important property of SA: it must offer a *virtually infinite* amount of space, so that one can perform as much reasoning as needed by scribing an unbounded (but finite) amount of statements[2].

Then as the name indicates, scribing some statement Φ on SA has the meaning of *asserting the truth of* Φ. It is thus an instance of the notion of *judgment* as identified by logicians like Frege and Martin-Löf, who would write it symbolically as $\vdash \Phi$. Naturally, the empty SA in interpreted as the absence of any assertion/judgment, and thus as vacuous truth $\top$.

Atoms Peirce often used sentences expressed in natural language as the most elementary statements scribed on SA, probably for pedagogical purposes. However he made clear that the informal meaning of these sentences, that is their denotation in the real world, is irrelevant to the process of pure logical deduction. This agrees with the modern view on *atomic* propositions, which are taken to be arbitrary abstract symbols drawn from some countably infinite set $\mathscr{A}$. We will use letters $a, b, c \ldots$ to denote such statements and call them *atoms*.

Juxtaposition Recall that one can scribe an arbitrary number of statements on SA, thus asserting the truth of each of them simultaneously. That is, *juxtaposition* has the meaning of *conjunction*, as we know from the introduction rule for $\wedge$ in natural deduction. However, symbolic connectives do not exist in the syntax of EGs, because Peirce aimed precisely for a

[2] Just like a Turing machine has an infinite tape, so that one can perform as much computation as needed. In symbolic logic, this is captured by the fact that formulas, although usually finite, can have an unbounded size.

symbolless — what he called *iconic* [24] — notation for logic. Thus very concisely,

$$\frac{\vdash \Phi \quad \vdash \Psi}{\vdash \Phi \land \Psi} \land\mathsf{i} \qquad \text{is expressed by the graph} \qquad \Phi \quad \Psi$$

Note that in symbolic logic, Φ and Ψ can be arbitrarily complex formulas, not just atoms. In EGs, a complex statement — that we will call a *graph* — is any delimited portion/area of SA, as long as the delimitation does not interrupt the continuity of some *token*.

Scroll In the fragment of EGs considered in this article, a token[3] is a scribed occurrence of either an atom, or what Peirce called a *scroll*. In the words of Peirce himself [19, pp. 533–534]:

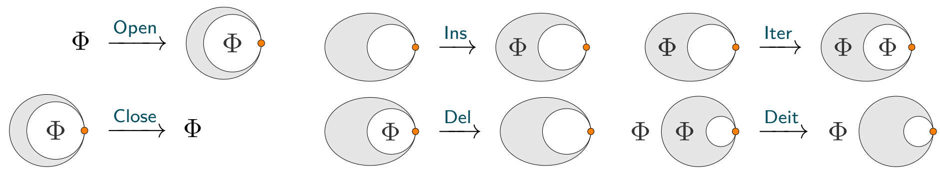
> Accordingly, since logic has primarily in view argument, and since the conclusiveness of an argument can never be weakened by adding to the premisses or by subtracting from the conclusion, I thought I ought to take the general form of argument as the basal form of composition of signs in my diagrammatization; and this necessarily took the form of a "scroll", that is [...] a curved line without contrary flexure and returning into itself after once crossing itself, and thus forming an outer and an inner "close".

Examples of scrolls can be found in Figure 1, where the curved line is depicted by a white ellipse (the "inner close") nested in a gray ellipse (the "outer close"), with a unique orange intersection point emphasizing the "once crossing itself" part. Peirce also used the term *sep* instead of "close" to refer to any of the two closed curves that make up a scroll[4]. Following Pietarinen [11], we will use the terms *inloop* and *outloop* to designate the inner and outer seps of a scroll, respectively. By "general form of argument", Peirce alludes to logical *implication* $\Phi \Rightarrow \Psi$, where the antecedant Φ resides in the outloop, and the consequent Ψ in the inloop. Φ and Ψ can be arbitrarily complex graphs, so that in particular a scroll located in the outloop of another scroll will be drawn with its gray and white shading inverted in order to reflect the change of polarity (see for instance Figure 3a).

Finally, note that in classical logic seps can be interpreted as *negations* because of the classical equivalence $\Phi \Rightarrow \Psi \simeq \neg(\Phi \land \neg\Psi)$. Indeed EGs were invented around 1896 before the advent of intuitionistic logic, so that Peirce had a boolean interpretation of logical operations in mind. However in this article, we will demonstrate that the inference rules of EGs are not only intuitionistically sound (when restricted to scrolls), but that the notion of computation that emerges from them is very much in line with the modern conception of constructivism.

Illative transformations Peirce called the inference rules of EGs *illative transformations*. They are traditionally understood as *rewriting rules*, similarly to the rules of string diagram calculi in category theory, or to those of the calculus of structures in deep inference [6]. One can find various presentations in the literature, some of the rules being *equational* in nature (bidirectional) because of the equivalence between premiss and conclusion, the others being necessarily *oriented* (unidirectional). Here we stick to a fully oriented presentation, for reasons that will become clear in the next section. This makes for a total of six rules, illustrated with representative examples in Figure 2. They should be read from left to right as expressing *forward reasoning* from premiss to conclusion.

[3] Peirce used the word "token" as a synonym of the more contemporary term "occurrence", and the word "type" to refer to the common pattern that is instantiated in multiple occurrences of the same type.
[4] This is because a closed curve literally *sep*arates SA into two distinct areas, which is now known as the *Jordan curve theorem* in topology.



(a) Interaction rules **(b)** Argumentation rules

Figure 2 Dynamic illative transformations

While still informal, the following description specifies the rules in their full generality. Note that to avoid any ambiguity — and to foster the parallel with natural deduction — we will use the terms *introduction* and *elimination* to refer respectively to the act of *scribing* and *erasing* a graph from SA. We will also refer to white/gray-shaded areas — or alternatively to areas enclosed in an even/odd number of seps — as *positive*/*negative*.

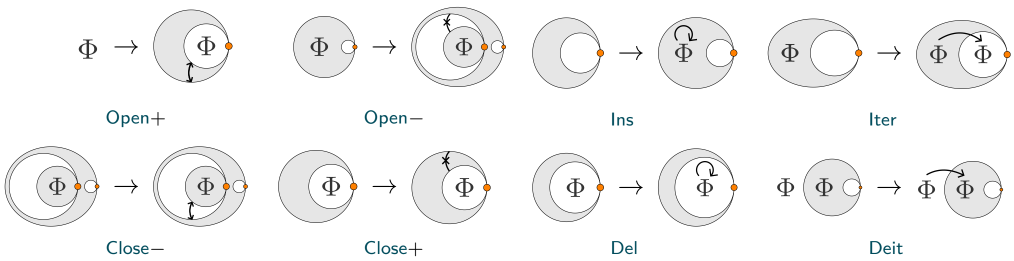
***Opening* (Open)** A scroll with empty outloop can be introduced around any graph Φ.
***Closing* (Close)** Any scroll with empty outloop can be eliminated.
***Insertion* (Ins)** Any graph Φ can be introduced in a negative location.
***Deletion* (Del)** Any graph Φ can be eliminated from a positive location.
***Iteration* (Iter)** Any graph Φ can be introduced in a positive location, as long as Φ already occurs in the area of a sep that contains said location.
***Deiteration* (Deit)** Any graph Φ can be eliminated from a negative location, as long as Φ already occurs in the area of a sep that contains said location.

A remarkable feat of Peirce's rules — on which he insisted very much — is that they are only expressed in terms of introductions (first row) and eliminations (second row) of graphs on SA. We will refer to this property as *illative atomicity*. Indeed, Peirce thought that those were the *smallest* steps in which reasoning could be dissected, making his system extremely appropriate for *analytical* purposes. This is summarized in the following excerpt [19, p. 533]:

> In the first place, the most perfectly analytical system of representing propositions must enable us to separate illative transformations into indecomposable parts. Hence, an illative transformation from any proposition, A, to any other, B, must in such a system consist in first transforming A into AB, followed by the transformation of AB into B. For an omission and an insertion appear to be indecomposable transformations and the only indecomposable transformations.

We qualify the first two rules Open and Close as *interaction* rules. Indeed from a game-semantical point of view, the Open rule starts an interaction with the opponent by introducing a negative (resp. positive) outloop in a positive (resp. negative) area, while the Close rule ends this interaction by erasing the scroll while keeping the inloop's content. They are obviously sound in both intuitionistic and classical logic, as they correspond to the equivalence $\top \Rightarrow \Phi \simeq \Phi$.

We qualify the other rules as *argumentation* rules, following Peirce's description of the scroll as the "general form of argument". They generalize exactly the *structural* rules found in sequent calculus and the calculus of structures (as well as the *switch* rule in the latter), because they can be applied in locations of arbitrary depth and polarity, not just at the top-level of sequents or in positive contexts.



(a) Interaction rules **(b)** Argumentation rules

Figure 3 Static illative transformations

The Ins and Del rules correspond respectively to *weakening* and *coweakening*. They capture the monotonicity of entailment, expressed by Peirce in the sentence "[...] the conclusiveness of an argument can never be weakened by adding to the premisses [(Ins rule)] or by subtracting from the conclusion [(Del rule)] [...]".

Lastly, the (de)iteration rules Iter and Deit can be seen respectively as generalizations of the *cocontraction* and *contraction* rules. They capture when some source occurrence of a statement Φ *justifies* a distinct target occurrence of the same Φ, where *justification* consists in either adding a positive conclusion (Iter) or removing a negative assumption (Deit). Their soundness is less straightforward than for other rules, and for lack of space we rely on secondary sources like [2] where it has been proved formally with respect to intuitionistic Kripke semantics, albeit in a slightly different setting.

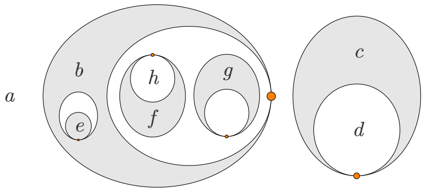
3 Reifying inference

Justifications In all presentations of illative transformations found in secondary literature, iteration and deiteration are *polarity-independent*: they can each justify statements in both positive and negative locations. Then deiteration can be seen as the converse of iteration, and since the premiss and conclusion are equivalent, they are usually considered as a single equational rule. However, Peirce himself noted that this presentation is "[...] valid, sufficient for its purpose, and convenient in practice, [...][but there is a] more scientific way [to proceed]" ([19, p. 536]). Then he goes on to explain this "more scientific way", of which one clause reads as follows [19, p. 537]:

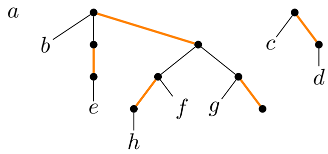
> [...] if Ω be a **recto [(i.e. positive)]** area, any simple Graph already scribed upon Y may be iterated upon Ω; while if Ω be a **verso [(i.e. negative)]** Area, any simple Graph already scribed upon Y and iterated upon Ω may be deiterated by being deleted or abolished from Ω.

This subtle polarity restriction turns out to be very important in order to *reify* (de)iteration in the syntax of EGs. Indeed, recall that our goal is to find a way to *record* illative transformations on SA as they are being performed, so that the result of a sequence of such transformations is the static proof object that was built, rather than its mere conclusion; just as the conclusion of a derivation in type theory is a judgment that holds not only a type, but also a term witnessing this type.

Then, since a (de)iteration justifies a target occurrence of some graph Φ with a source occurrence of Φ, it is tempting to represent it by a simple *arrow* with corresponding source and target. This gives the *static* Iter and Deit rules of Figure 3b. Note that since we want all transformations to strictly *add* information incrementally, the justified occurrence of Φ in Deit



(a) Topological



(b) Combinatorial

Figure 4 Topological and combinatorial representations of a simple scroll structure. In Fig. 4b, labelled leaves are atoms, •-nodes are seps, and orange edges are attachments of inloops to outloops.

is not eliminated anymore. Then the only way to distinguish between these static versions of Iter and Deit is to look at the polarity of the target location.

The same principle can be applied to the Ins and Del rules, which are now represented by *looping arrows*. The intuition is that a graph introduced in a negative area with the Ins rule corresponds to an *assumption*, which is *self-justified* in the sense that the reasoner decides it does not require further justification (avoiding the well-known infinite regression problem). Dually, a graph eliminated from a positive area with the Del rule *annihilates itself*, capturing the reasoner's intent to prevent further usage of the knowledge about this graph's truth.

Interactions There is a certain *homeomorphic* flavor to interaction rules, expressed by Peirce for the Close rule as follows: "[...] the two walls [(i.e. seps)] of the scroll, when nothing is between them, fall together, **collapse**, disappear, and leave only the contents of the inner close standing [...]" ([19, p. 534]). We diagrammatize this "collapse" deformation by drawing two arrows departing from each sep and meeting at their tips, which also gives the impression of a *cross* symbolizing elimination (rule Close+ in Figure 3a). Dually, opening (rule Open+) is symbolized with a double-ended arrow whose tips touch the two seps, evoking an *expansion* movement as if SA was teared apart to create negative space. In order to stay consistent with the polarity-dependent interpretation of arrows in argumentation rules, collapse (resp. expansion) arrows are interpreted as opening (resp. closing) transformations in negative areas, giving two new variants Open− and Close− of the rules.

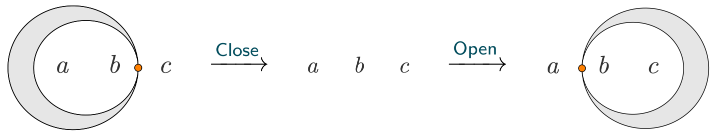
4 Combinatorial scroll nets

Scroll structures As noted by various authors[5], the nesting of seps on SA induces a *forest* structure on EGs: each sep constitutes a node, whose children are either leaves corresponding to atoms or empty seps residing in the area of the sep, or nodes corresponding to nested non-empty seps. One also needs a way to keep track of the attachment of inloops to outloops: Figure 4b illustrates how this can be achieved by coloring edges that relate an inloop to its parent outloop in orange. This gives the following formal definition:

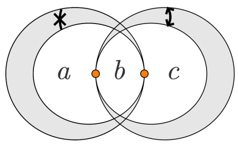
▶ **Definition 1** (Scroll structure). *A simple scroll structure is a triple* $\Phi = \langle G, \ell, \infty \rangle$ *where:*

- $G = \langle V, \rightarrowtriangle \rangle$ *is a finite rooted out-forest (directed from roots to leaves) with edge set* $\rightarrowtriangle \subseteq V \times V$*. We write* $v \rightarrowtriangle u$ *for* $(v, u) \in \rightarrowtriangle$*, and* $\rightarrowtriangle^*$ *for the transitive closure of* $\rightarrowtriangle$*.*
- $\ell : L \rightharpoonup \mathcal{A}$ *is a partial function labelling the leaves* $L \subseteq V$ *with atoms.*

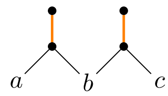
[5] See for instance the Tree Existential Graphs of Roberts and Pronovost [22], or [1, Section 2.2].



(a) Dynamic proof trace



(b) Static proof object



(c) Scroll structure with sharing

Figure 5 Example of proof with sharing inloops

- $\multimap \subseteq \rightarrow$ *is a subset of* attachments *satisfying the following well-formedness conditions:*
  1. *(Atoms are not inloops) If* $u \multimap v$ *then* $v \notin \mathrm{dom}(\ell)$*.*
  2. *(Every sep is attached)* $\forall v \in V \setminus \mathrm{dom}(\ell)$, $\exists! u \in V$ *such that either* $u \multimap v$ *or* $v \multimap u$*.*

We will use letters Φ, Ψ, Ξ to range over simple scroll structures, subscripting components accordingly for disambiguation (e.g. V_Φ for the vertices of Φ). The adjective "simple" is here meant as a reference to *simple type theory*, as will become clear in Section 6; we will usually omit it for conciseness.

▶ **Definition 2** (Polarity). *We say that a node* $v \in V_\Phi$ *is* positive *(resp.* negative*) if its distance from a root is even (resp. odd). We denote the sets of* positive *and* negative *nodes of* Φ *by* V_Φ^+ *and* V_Φ^-*, respectively.*

Scroll structures capture Peirce's informal notion of graph built out of atoms and scrolls, and thus the structure of statements in implicative logic. However they cannot express the *sharing* structure of some proofs in EGs. Figure 5a shows a proof trace for an analogue of the associativity of conjunction, where grouping of conjuncts is achieved by enclosing them in scrolls with empty outloops, instead of the traditional symbolic device of parentheses. If one wants to represent faithfully in the corresponding scroll net the identity of the two occurrences of b in the premiss and conclusion, the only way is to have the closed and opened scrolls overlap on their inloop, as illustrated in Figure 5b. Then b has two parents in the scroll structure (Figure 5c), because the two inloops *share* it as a common child. This means that we need to relax forests into *directed acyclic graphs* (*DAGs*):

▶ **Definition 3** (Sharing). *A* simple scroll structure with sharing Φ *(or* simple SSS *for short) is the same data as a simple scroll structure, except that the forest becomes a DAG with the constraint that if a node has more than 1 parent, then all its parents must be inloops. Formally, if* $u \rightarrow v$ *and* $u' \rightarrow v$ *with* $u \neq u'$*, then* $\exists u_0, u_0'$ *such that* $u_0 \multimap u$ *and* $u_0' \multimap u'$*.*

The additional constraint is here justified by the fact that it does not make sense for two distinct outloops to overlap or share content, because the Open and Close rules only work on scrolls with empty outloops.

Scroll nets The last step consists in encoding graph-theoretically the various types of arrows introduced in Figure 3 to represent statically illative transformations. We again make the distinction between argumentation and interaction rules, which act respectively on the nodes and edges of a SSS:

▶ **Definition 4** (Scroll net). *A simple scroll net is a triple* $\mathfrak{S} = \langle \Phi, \mathcal{A}, \mathcal{I} \rangle$ *where:*

- Φ *is a simple SSS.*
- $\mathcal{A} = \langle \curvearrowright, \circlearrowleft \rangle$ *is a pair of a directed forest of justifications* $\curvearrowright \subseteq V_\Phi \times V_\Phi$ *and a set of self-justifications* $\circlearrowleft \subseteq V_\Phi$ *called the argumentation of* $\mathfrak{S}$*. We write respectively* $u \curvearrowright v$ *and* $\circlearrowleft u$ *when* $(u, v) \in \curvearrowright$ *and* $u \in \circlearrowleft$*.*
- $\mathcal{I} = \langle \longleftrightarrow, \nrightarrow \rangle$ *is a pair of an expansion and a collapse* $\longleftrightarrow, \nrightarrow \subseteq \propto_\Phi$ *called the interaction of* $\mathfrak{S}$*. We write* $v \nleftrightarrow u$ *as a shorthand when both* $v \longleftrightarrow u$ *and* $v \nrightarrow u$*.*

We will use letters $\mathfrak{S}, \mathfrak{T}, \mathfrak{U}$ to range over scroll nets, again subscripting components accordingly when disambiguation is necessary, so that for instance $v \curvearrowright_{\mathfrak{S}} u$ expresses that node v justifies node u in $\mathfrak{S}$. Now, the choice of a directed *forest* of justifications rather than an arbitrary digraph requires some explanations. Compared to an arbitrary digraph, a forest must satisfy additionally both *acyclicity* and *unicity of parents*:

Unique parents A priori, a node could be the target as well as the source of an arbitrary number of justifications. But upon closer inspection, it appears that while it makes sense to be the source of many arrows — the same statement can justify multiple copies of itself, it is impossible to be the target of more than one justification. Indeed in the dynamic understanding of (de)iteration (Figure 2b), the justified node is either introduced or eliminated, and there is no sense in which the exact same node could be introduced or eliminated more than once. Said differently, a node can only be (de)duplicated from a single source node.

Acyclicity Here the formal reasons are much more subtle, although they are also related to the dynamic reading of illative transformations; we conjecture that the latter indeed preserve acyclicity. Intuitively though, it is clear that one wants to avoid cyclic justifications in reasoning: there is no meaning in asserting that "u is true because v is true because u is true because v is true because...". Acyclicity is also a fundamental component of correctness criterions for most graphical proof formalisms like proof nets, combinatorial proofs and expansion tree proofs [13].

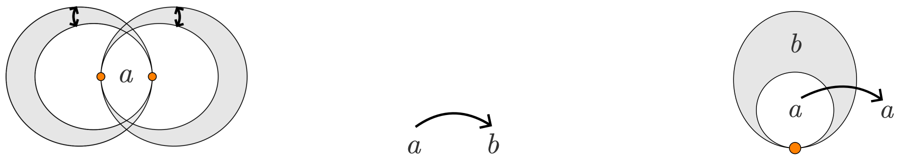
Boundaries Given a scroll net, it is not immediately obvious *what* it is proving — i.e. what are its *premiss* and *conclusion*, because both are "superposed" in a non-trivial way. For instance in the scroll nets of Figures 1b and 5b, the same atomic node b occurs both in the premiss and the conclusion. This is actually a powerful feature, as it makes proofs much more compact by identifying occurrences that would appear as separate copies in most other proof formalisms, including proof nets and combinatorial proofs. Fortunately there is a simple algorithm for disentangling scroll nets, by exploiting illative atomicity and duality. First we recall some standard graph-theoretic notions:

▶ **Definition 5.**
The subgraph of $\mathfrak{S}$ reachable from v *is defined as* $v{\downarrow} = \{(u, w) \mid v \multimap^* u \text{ and } u \multimap w\}$*. The sets of* parents *and* children *of* v *are defined respectively as* $\rightrightarrows v = \{u \mid u \multimap v\}$ *and* $v \rightrightarrows = \{u \mid v \multimap u\}$*. We say that* v *is a sibling of* u*, written* $v \bowtie u$*, if* $\rightrightarrows v = \rightrightarrows u$*.*

Then we define operations that update a scroll structure by adding or removing nodes and edges in its DAG:

▶ **Definition 6.**

- *The pruning* $\mathsf{prune}(\Phi, v)$ *of a node* $v \in V_\Phi$ *is the result of removing* $v{\downarrow}$ *and all edges* $u_i \multimap v$ *from* G_Φ*, updating other components of* Φ *accordingly.*
- *The collapsing* $\mathsf{collapse}(\Phi, v)$ *of a scroll* $v \propto_\Phi u$ *is the result of pruning every* $w \in v \rightrightarrows \setminus \{u\}$*, removing* v*,* u *and their associated edges from* G_Φ*, and adding an edge* $v_i \multimap u_j$ *for every* $v_i \in \rightrightarrows v$ *and* $u_j \in u \rightrightarrows$*.*



(a) Overlapping **(b)** Identity **(c)** Scoping

Figure 6 Examples of incorrect scroll nets

We also need to characterize when a node is either *introduced*/*eliminated* or *opened*/*closed* dynamically by some illative transformation in the argumentation or interaction of a scroll net. As illustrated in Figure 3, this can be done by just looking at the *polarity* of nodes:

▶ **Definition 7** (Edit state). *The sets of opened, closed, introduced and eliminated nodes of a scroll net $\mathfrak{S}$ are defined respectively by:*

$$\begin{aligned}
\mathsf{Opn}(\mathfrak{S}) &= \{v \mid \exists u.(v \in V^+ \wedge v \longleftrightarrow u) \vee (v \in V^- \wedge v \nleftrightarrow u)\} \\
\mathsf{Clo}(\mathfrak{S}) &= \{v \mid \exists u.(v \in V^- \wedge v \longleftrightarrow u) \vee (v \in V^+ \wedge v \nleftrightarrow u)\} \\
\mathsf{Intro}(\mathfrak{S}) &= \{v \mid (v \in V^+ \wedge \exists u.u \curvearrowright v) \vee (v \in V^- \wedge \circlearrowleft v)\} \\
\mathsf{Elim}(\mathfrak{S}) &= \{v \mid (v \in V^- \wedge \exists u.u \curvearrowright v) \vee (v \in V^+ \wedge \circlearrowleft v)\}
\end{aligned}$$

We can now define straightforwardly the *boundaries* of a scroll net:

▶ **Definition 8** (Boundaries). *The boundaries of a scroll net $\mathfrak{S}$ are two simple SSSs:*
- *the premiss* $\lceil \mathfrak{S} \rceil = \mathsf{collapse}(\mathsf{prune}(G_{\mathfrak{S}}, \mathsf{Intro}(\mathfrak{S})), \mathsf{Opn}(\mathfrak{S}))$*;*
- *the conclusion* $\lfloor \mathfrak{S} \rfloor = \mathsf{collapse}(\mathsf{prune}(G_{\mathfrak{S}}, \mathsf{Elim}(\mathfrak{S})), \mathsf{Clo}(\mathfrak{S}))$*.*

Note that we apply first pruning and then collapsing on *sets* of nodes, abstracting from the particular order in which individual operations are done. We observe that this does not pose any problem in practice, because this process should always be *confluent*. In fact, we could also first perform collapsing before pruning because a node in a correct scroll net cannot be both opened and introduced or both closed and eliminated, i.e. $\mathsf{Opn}(\mathfrak{S}) \cap \mathsf{Intro}(\mathfrak{S}) = \varnothing$ and $\mathsf{Clo}(\mathfrak{S}) \cap \mathsf{Elim}(\mathfrak{S}) = \varnothing$. One can easily convince oneself that the algorithm works by applying it to the various examples of scroll nets presented earlier, where the premiss and conclusion should match respectively the first and last scroll structure in the corresponding proof trace.

It is then possible to characterize *incomplete* or *partial* scroll nets as those with a non-empty premiss, just as partial derivations in other formalisms are those where some leaves are not closed by a nullary rule:

▶ **Definition 9** (Completeness). *A scroll net $\mathfrak{S}$ is said to be complete if $\lceil \mathfrak{S} \rceil = \varnothing$ with $\varnothing$ denoting the empty scroll structure $\langle\langle \varnothing, \varnothing \rangle, \varnothing, \varnothing\rangle$, otherwise it is incomplete.*

5 Correctness

Incorrectness In practice, we want the boundaries of a scroll net to be plain scroll structures without inloop sharing, so that they can be interpreted straightforwardly as logical statements. However, this is only guaranteed for *correct* scroll nets that can be built by applying a sequence of illative transformations, since there is no transformation that makes two inloops overlap.

More generally, Definition 4 is not restrictive enough to capture only the class of correct scroll nets. Figure 6 shows three examples of incorrect scroll nets: in Fig. 6a it is precisely because of the overlap problem just mentioned; in Fig. 6b it comes from the source and target of a justification not being identical; lastly in Fig. 6c, it is caused by a justification that violates scoping by going out of a scroll that is neither opened nor closed.

Although there is probably some geometric criterion that could capture each of these mistakes by analyzing solely the static structure of scroll nets, in this paper we only formalize the baseline *sequential* correctness criterion. Informally, we want to say that a scroll net $\mathfrak{S}$ is correct if there exists a total ordering of all elements in its argumentation and interaction such that the corresponding dynamic sequence of illative transformations builds exactly $\mathfrak{S}$.

Derivations To formalize this intuition, we need to define additional operations on scroll nets that capture precisely the *incremental* application of illative transformations, where the latter are recorded in the argumentation and interaction. Following standard terminology, we call these operations *derivation* rules and define them in Figure 7. They are the fully general version of the more visual examples provided in Figure 3. Although we use the inference line notation, here we are not building derivation trees: premisses correspond to conditions that the input $\mathfrak{S}$ must satisfy for the rule to be applicable, while the conclusion $\mathfrak{S}'$ is the scroll net which results from applying the rule. Thus every rule R can be seen as a partial map of scroll nets. To express the rules as compactly as possible, we rely on the following notational conventions:

- we omit to put $\mathfrak{S}$ in subscript when referring to its components, and avoid outer brackets $\langle\rangle$ for the triple defining the conclusion $\mathfrak{S}'$;
- $X \uplus Y$ denotes the *disjoint union* of two sets. If $x \in X$ and $y \in Y$, then $x^{\triangleleft} \in X \uplus Y$ and $y^{\triangleright} \in X \uplus Y$ denote the *left* and *right copy* of the original elements; we also write $X \cup x$ as a shorthand for $X \cup \{x\}$;
- $\Phi \simeq \Phi'$ denotes the existence of an *isomorphism* of scroll structures, i.e. an isomorphism between underlying DAGs/forests that preserves the atom labels and attachments.
- We write $\mathfrak{S} \triangleright \mathfrak{T}$ to express that there is some rule R such that $\mathfrak{T} = \mathsf{R}(\mathfrak{S})$, and $\mathfrak{S} \triangleright^* \mathfrak{T}$ to denote the transitive closure of $\triangleright$, or equivalently that there exists a sequence of rules $\mathsf{R}_1, \ldots, \mathsf{R}_n$ such that $\mathfrak{T} = \mathsf{R}_n \circ \ldots \circ \mathsf{R}_1(\mathfrak{S})$.

Soundness We now state a series of lemmas whose proofs would ensure the adequacy of our formal definition of derivation rules. Again for lack of space, we omit such proofs. First we want to ensure that if $\mathfrak{S}$ is a scroll net and $\mathfrak{S} \triangleright \mathfrak{T}$, then $\mathfrak{T}$ is still a well-formed scroll net:

▶ **Lemma 10** (Well-formedness preservation). *If $\mathfrak{S} \triangleright \mathfrak{T}$ then the following holds:*
- *$G_{\mathfrak{T}}$ is a DAG satisfying the constraint of Definition 3;*
- *$\backsim_{\mathfrak{T}} \subseteq \twoheadrightarrow_{\mathfrak{T}}$ and it satisfies the well-formedness conditions of Definition 1;*
- *$\curvearrowright_{\mathfrak{T}}$ is a forest.*

Then we want to ensure that derivation rules do not change the boundaries of a scroll net, which is akin to *subject reduction* in type theory:

▶ **Definition 11.** *We say that $\mathfrak{S}$ is interpretable iff $G_{\lceil \mathfrak{S} \rceil}$ and $G_{\lfloor \mathfrak{S} \rfloor}$ are forests.*

▶ **Lemma 12** (Subject construction). *If $\mathfrak{S} \triangleright \mathfrak{T}$ then the following holds:*
- ***(Premiss preservation)*** $\lceil \mathfrak{S} \rceil \simeq \lceil \mathfrak{T} \rceil$.
- ***(Interpretability)*** *if $\mathfrak{S}$ is interpretable, so is $\mathfrak{T}$.*

We can then show the following results about logical soundness:

▶ **Definition 13** (Interpretation). *The interpretation* $[\![\Phi]\!]$ *of a scroll structure* Φ *is a formula defined by induction on the depth of* G_Φ*:*

$$[\![\varnothing]\!] = \top \qquad [\![v_1, \ldots, v_n]\!] = [\![v_1]\!] \wedge \ldots \wedge [\![v_n]\!]$$

$$[\![v]\!] = \begin{cases} a & \text{if } v^{\rightrightarrows} = \varnothing \text{ and } \ell(v) = a \\ [\![v_1, \ldots, v_n]\!] \Rightarrow [\![w_1, \ldots, w_m]\!] & \text{if } v^{\rightrightarrows} = \{v_1, \ldots, v_n, u\},\ u \propto w \text{ and } w^{\rightrightarrows} = \{w_1, \ldots, w_m\} \end{cases}$$

We write $\lceil\!\lceil \mathfrak{S} \rceil\!\rceil$ *and* $\lfloor\!\lfloor \mathfrak{S} \rfloor\!\rfloor$ *as a shorthand for* $[\![\lceil \mathfrak{S} \rceil]\!]$ *and* $[\![\lfloor \mathfrak{S} \rfloor]\!]$.

▶ **Lemma 14.** *If* $\Phi \simeq \Psi$ *then* $[\![\Phi]\!] \simeq [\![\Psi]\!]$*, i.e.* $[\![\Phi]\!]$ *and* $[\![\Psi]\!]$ *are logically equivalent.*

▶ **Lemma 15** (Conclusion entailment). *If* $\mathfrak{S} \rhd \mathfrak{T}$ *and* $\mathfrak{S}$ *is interpretable, then* $\lfloor\!\lfloor \mathfrak{S} \rfloor\!\rfloor \vdash \lfloor\!\lfloor \mathfrak{T} \rfloor\!\rfloor$.

▶ **Theorem 16** (Soundness). *If* $\Phi \rhd^* \mathfrak{S}$ *then* $[\![\Phi]\!] \vdash \lfloor\!\lfloor \mathfrak{S} \rfloor\!\rfloor$.

▶ **Corollary 17.** *If* $\varnothing \rhd^* \mathfrak{S}$ *then* $\vdash \lfloor\!\lfloor \mathfrak{S} \rfloor\!\rfloor$.

Finally, the above soundness results legitimate our sequential definition of correctness:

▶ **Definition 18** (Correctness). *We say that a scroll net* $\mathfrak{S}$ *is correct iff* $\mathfrak{S}$ *is interpretable and* $\lceil \mathfrak{S} \rceil \rhd^* \mathfrak{S}$.

Composition A first very natural way to compose two scroll nets $\mathfrak{S}$ and $\mathfrak{T}$ is by taking their *juxtaposition*. As usual with graphical formalisms, this is defined by taking the *disjoint union* of their respective components:

▶ **Definition 19** (Horizontal composition). *The horizontal composition of two scroll structures* Φ *and* Ψ *and two scroll nets* $\mathfrak{S}$ *and* $\mathfrak{T}$ *are defined by*

$$\Phi \uplus \Psi = \langle G_\Phi \uplus G_\Psi, \ell_\Phi \uplus \ell_\Psi, \propto_\Phi \uplus \propto_\Psi \rangle$$

$$\mathfrak{S} \uplus \mathfrak{T} = \langle \Phi_\mathfrak{S} \uplus \Phi_\mathfrak{T}, \langle \curvearrowright_\mathfrak{S} \uplus \curvearrowright_\mathfrak{T}, \circlearrowright_\mathfrak{S} \uplus \circlearrowright_\mathfrak{T} \rangle, \langle \longleftrightarrow_\mathfrak{S} \uplus \longleftrightarrow_\mathfrak{T}, \mathrel{-\!\!\times\!\!-}_\mathfrak{S} \uplus \mathrel{-\!\!\times\!\!-}_\mathfrak{T} \rangle \rangle$$

The boundaries of $\mathfrak{S} \uplus \mathfrak{T}$ will be interpreted as the *conjunction* of the boundaries of $\mathfrak{S}$ and $\mathfrak{T}$, i.e. $\lceil\!\lceil \mathfrak{S} \uplus \mathfrak{T} \rceil\!\rceil \simeq \lceil\!\lceil \mathfrak{S} \rceil\!\rceil \wedge \lceil\!\lceil \mathfrak{T} \rceil\!\rceil$ and $\lfloor\!\lfloor \mathfrak{S} \uplus \mathfrak{T} \rfloor\!\rfloor \simeq \lfloor\!\lfloor \mathfrak{S} \rfloor\!\rfloor \wedge \lfloor\!\lfloor \mathfrak{T} \rfloor\!\rfloor$. Importantly, it should also be the case that $\mathfrak{S} \uplus \mathfrak{T}$ is correct whenever $\mathfrak{S}$ and $\mathfrak{T}$ are. Indeed, the intuition is that the two derivations $\lceil \mathfrak{S} \rceil \rhd^* \lfloor \mathfrak{S} \rfloor$ and $\lceil \mathfrak{T} \rceil \rhd^* \lfloor \mathfrak{T} \rfloor$ can be performed in parallel to yield a derivation $\lceil \mathfrak{S} \rceil \uplus \lceil \mathfrak{T} \rceil \rhd^* \lfloor \mathfrak{S} \rfloor \uplus \lfloor \mathfrak{T} \rfloor$. This kind of composition is typical of *deep inference* formalisms like the calculus of structures or open deduction [28].

The more standard kind of composition corresponds to the cut rule in sequent calculus or to the composition of morphisms in categorical semantics, and we call it *vertical composition*.

▶ **Definition 20** (Compatibility). *We say that two scroll nets* $\mathfrak{S}$ *and* $\mathfrak{T}$ *are compatible, written* $\mathfrak{S} \sim \mathfrak{T}$*, whenever* $\lfloor \mathfrak{S} \rfloor \simeq \lceil \mathfrak{T} \rceil$.

▶ **Definition 21** (Superposition). *The superposition* $\mathfrak{S} \gtrdot \mathfrak{T}$ *of two correct and compatible scroll nets* $\mathfrak{S} \sim \mathfrak{T}$ *is defined as the lifting of the derivation* $\lfloor \mathfrak{S} \rfloor \simeq \lceil \mathfrak{T} \rceil \rhd^* \mathfrak{T}$ *into a derivation* $\mathfrak{S} \rhd^* \mathfrak{S} \gtrdot \mathfrak{T}$*, i.e. the same sequence of rules is applied modulo the isomorphism* $\lfloor \mathfrak{S} \rfloor \simeq \lceil \mathfrak{T} \rceil$.

▶ **Definition 22** (Vertical composition). *Given two correct and compatible scroll nets* $\mathfrak{S} \sim \mathfrak{T}$*, their vertical composition* $\mathfrak{T} \circ \mathfrak{S}$ *is defined by taking the respective derivations* $\lceil \mathfrak{S} \rceil \rhd^* \mathfrak{S}$ *and* $\lceil \mathfrak{T} \rceil \rhd^* \mathfrak{T}$*, and composing those into a derivation* $\lceil \mathfrak{S} \rceil \rhd^* \mathfrak{S} \gtrdot \mathfrak{T} = \mathfrak{T} \circ \mathfrak{S}$.

These definitions rely on the fact that one can choose *canonically* a derivation for a correct scroll net $\mathfrak{S}$, corresponding to our aforementioned intuition that there should always be a way to totally order the elements of $\mathcal{A}_{\mathfrak{S}} \uplus \mathcal{I}_{\mathfrak{S}}$ into the associated sequence of illative transformations. It is also not entirely clear that the so-called *superposition* operation is formally well defined. The dynamic intuition is that the nodes and edges that appear in $\lfloor \mathfrak{S} \rfloor$ are included in $\mathfrak{S}$, so that every rule applicable in $\lfloor \mathfrak{S} \rfloor$ can equally be applied in $\mathfrak{S}$ (what we called "lifting" in Definition 21). The static, geometric intuition is that $\mathfrak{S}$ and $\mathfrak{T}$ being compatible means that they have a common (modulo isomorphism) boundary, so that one can literally superpose the topological representations of $\mathfrak{S}$ and $\mathfrak{T}$ on this boundary to obtain $\mathfrak{S} \gtrdot \mathfrak{T}$. This can be visualized in Figures 1 and 5 by turning the proof traces into recording derivations, splitting said derivations into sub-derivations, and checking that their recombinations through superposition gives back the expected scroll net.

6 Computation

Detours In natural deduction, a so-called *detour* — corresponding to a *β-redex* in simply typed λ-calculus — arises when an introduction rule on some formula is followed by an elimination rule on the same formula. Similarly in scroll nets, we will call *detour* a node that is both introduced and eliminated, in the sense of being entirely scribed and then entirely erased from SA. A simple argument by combinatorial exhaustion shows that there are only 4 possible shapes of detours, depicted on the left-hand side in Figure 8. Indeed, a detour is always a scroll that falls under one of the following cases:

Interaction/Interaction ($\rightsquigarrow_{\mathsf{ii}}$) opened then closed, either in a positive or negative location;
Interaction/Argumentation ($\rightsquigarrow_{\mathsf{ia}}$) opened then deleted, or inserted then closed;
Argumentation/Interaction ($\rightsquigarrow_{\mathsf{ai}}$) iterated then closed, or opened then deiterated;
Argumentation/Argumentation ($\rightsquigarrow_{\mathsf{aa}}$) iterated then deleted, or inserted then deiterated.

Although we have given 8 cases, they can be divided in 4 pairs that have the same shape: it is then the polarity of the detour that determines in which order the two illative transformations have been performed. There is also a 9^{th} case when two argumentation rules interact on an atom, but we consider it as a variant of the previous argumentation/argumentation case to preserve symmetry.

Detour reduction In Figure 8, we give for each kind of detour a general *reduction rule* (subdivided in a scroll and atom case for $\rightsquigarrow_{\mathsf{aa}}$). Here we depict the detour in a positive area, but this also works by inverting polarities thanks to the aforementioned polarity-invariance. Currently these rules are experimental, although one can check that they correctly preserve the boundaries of the scroll net. To do so, it is necessary to observe the two following facts:

- the premiss and conclusion become inverted when inverting the polarity. This means that it is the premiss (resp. conclusion) of the antecedant of a scroll — which is itself a scroll net — that appears in the conclusion (resp. premiss) of the scroll;
- for any two composable scroll nets $\mathfrak{S}$ and $\mathfrak{T}$, $\lceil \mathfrak{S} \gtrdot \mathfrak{T} \rceil = \lceil \mathfrak{S} \rceil$ and $\lfloor \mathfrak{S} \gtrdot \mathfrak{T} \rfloor = \lfloor \mathfrak{T} \rfloor$.

Note that we depict a scroll net $\mathfrak{S}$ that appears inside another scroll net $\mathfrak{T}$ — i.e. a *subnet* of $\mathfrak{T}$ — by enclosing it in a two-part box labelled $\mathfrak{S}$ on the right, where the upper and lower parts contain respectively $\lceil \mathfrak{S} \rceil$ and $\lfloor \mathfrak{S} \rfloor$.

Simulating STLC Given the above detour reduction rules, it is now possible to simulate straightforwardly the simply typed λ-calculus. We express this translation diagrammatically in Figure 9, awaiting future work for a more rigorous graph-theoretic formalization. The translation is divided into two parts:

Static typing (Figure 9a) Each typing rule with premisses $\Gamma_i \vdash t_i : A_i$ for $1 \leq i \leq n$ and conclusion $\Gamma \vdash t : A$ is mapped to a scroll net $\mathfrak{S}$ such that $\lceil \mathfrak{S} \rceil$ (resp. $\lfloor \mathfrak{S} \rfloor$) is equal to (the translation of) Γ (resp. A), and where the inductive translations of sub-derivations appear as subnets $\mathfrak{S}_i$ with corresponding boundaries $\lceil \mathfrak{S}_i \rceil = \Gamma_i$ and $\lfloor \mathfrak{S}_i \rfloor = A_i$.

Dynamic computation (Figure 9b) Here we simulate the β-reduction rule with the two detour reduction rules $\rightsquigarrow_{\mathsf{aa}}$ and $\rightsquigarrow_{\mathsf{ii}}$. Note that this does not trigger any form of duplication or erasure, as would be the case with substitution in λ-calculus. We believe that substitution should happen when new detours in u and t generated by the application of the previous two rules are further reduced.

7 Conclusion

In this article we have laid out the foundations of the theory of scroll nets, focusing on its historical and conceptual genesis, its technical graph-theoretic formalization, and sketching its connection to the simply typed λ-calculus. As mentioned repeatedly, there is much room for improvement and development of the meta-theory of our formalism. In particular, we wish to find mathematical proofs for the various *correctness* results, as well as a more rigorous definition of the *superposition* operation $\gtrdot$. The latter should benefit from a *sequentialization theorem* formalizing the existence of a total ordering on the illative transformations of a scroll net. A deeper analysis of detours and detour elimination would also require an entire dedicated article. In the remainder, we summarize connections that the theory of scroll nets entertains with previous, related, and future works and applications that we envision.

Intuitionistic EGs In previous work [2], we have shown how to capture precisely provability in (full) intuitionistic first-order logic inside a variant of Oostra's system of intuitionistic EGs [17] dubbed *flower calculus*. Although it enjoys a nice metaphorical notation and some useful properties in the context of interactive theorem proving — such as *analyticity* and *invertibility* of all inference rules, it breaks the perfect duality stemming from the illative atomicity of rules found in the original approach of Peirce and Oostra, which was shown in this paper to be essential to the application of the Curry-Howard methodology to EGs.

Generalized scroll In [17], Oostra introduces a *horizontal* generalization of the scroll where it can have an arbitrary number n of inloops, subsuming seps as the case where $n = 0$. In our formalization, this would correspond to the possibility of having n *sibling* inloops $v_1 \bowtie \ldots \bowtie v_n$ in the same scroll u, i.e. $u \propto v_1 \ldots v_n$. Following the classical reading of the scroll as nested negations, one naturally interprets this construct through De Morgan equivalences by taking the *disjunction* of inloops, and the aforementioned works show how one can still capture intuitionistic logic in this setting. We have observed in preliminary work that one can also consider a *vertical* generalization by allowing chains of attachments of the form $v_1 \propto \ldots \propto v_m$, leading to a notion of (n, m)-scroll. Chains of attachments are naturally interpreted as *intuitionistic subtraction*, which should enable a novel treatment of *dual-intuitionistic* and *bi-intuitionistic* logic.

Classical logic Peirce's original system of EGs for propositional logic was called Alpha, and it captured *classical* logic by adopting the classical reading of the scroll. A straightforward way to adapt scroll nets to this setting would consist in dropping the attachments $\propto$ in

Definition 1, just as Peirce ignored them. However since seps can be seen as $(0,0)$-scrolls, we believe that a more general treatment would keep the notion of attachment, and instead characterize classical proofs as those where (de)iterations do not preserve *continuity*; that is, where an attached inloop can be duplicated into an unattached sep. Once the right framing of classical logic has been found, we believe it could provide interesting insights into the problem of finding a good notion of *proof identity* in classical logic [26], as well as a decomposition of the computational behavior of classical systems in the Curry-Howard tradition like the $\lambda\mu$-calculus [18] and System L [16].

Other logics Peirce had also devised extensions of Alpha that capture *first-order predicate* logic (Beta) and somewhat more speculatively *modal* and *higher-order* logics (Gamma), thus providing natural venues for extensions of scroll nets to these more expressive logics. Our discovery of the computational content of EGs and its close connection to λ-calculus should enable a *type-theoretic* approach to modal and higher-order logics, hopefully shedding light on both theory and applications to programming and interactive theorem proving.

Combinatorial proofs Intuitionistic combinatorial proofs have been recently introduced by Heijltjes et al. as "a concrete geometric semantics of intuitionistic logic" [7]. They also benefit from a computational interpretation [8] and have been related to game semantics [7]. However they exhibit a quite different graph-theoretic structure, with a strict separation between formulas in the arena and formulas in the game. In contrast, nodes in the scroll structure and argumentation/interaction of a scroll net are shared, leading to a more compact representation. Like scroll structures, arenas identify formulas that are equivalent modulo commutativity and associativity of conjunction. But contrary to scroll structures, they also conflate formulas equivalent modulo currying, which makes them unable to express computations under nested abstractions.

Bigraphs Scroll nets are closely related to the notion of *bigraph* introduced by Milner as a model of mobile interaction [14]. A bigraph consists of two independent structures: a *topograph* or *place graph* encoding spatial information as a forest, and a *monograph* or *link graph* encoding the connectivity of nodes as a hypergraph sharing the same vertices as the topograph. They have been generalized along two independent axes: the *bigraphs with sharing* of [23] relax the topograph forests into DAGs to represent overlapping of locations; and the *directed bigraphs* of [5] orient the monograph's edges to encode resource dependencies or information flow. Scroll structures have the same structure as topographs in bigraphs with sharing, while (static) illative transformations seem to be special cases of monographs in directed bigraphs. It is remarkable that scroll nets combine these two generalizations of a very recent model of computation, while being based entirely on the principles of EGs that predate both proof theory and the advent of computer science. It would be interesting to explore the possibility of formalizing illative transformations and detour reduction rules as a *bigraphical reactive system*, possibly in a categorical setting such as adhesive categories [10].

Proof script vs proof term We mentioned in the introduction the distinction between the notions of proof script and proof term in the interface of state-of-the-art ITPs like Rocq and Lean, which we coined more generally as that between proof objects and proof traces. This distinction seems to be essential to their successful usage in large-scale formalization efforts, but is unfortunately not reflected in current proof-theoretical frameworks. While proof term languages are based on dependent type theory and thus benefit from strong and (relatively) uniform theoretical foundations, proof scripts are expressed in a variety of metaprogramming and domain-specific languages with somewhat ad hoc structure and semantics[6]. This has led to a fragmented landscape of user interfaces for ITPs that limits their accessibility and interoperability, as well as the common belief that there is unavoidable accidental complexity in the various processes (e.g. type inference, proof search, elaboration, macro processing, pretty-printing) that bridge the gap between low-level terms and high-level tactics/notations.

We believe that scroll nets could provide a unified foundation for representing both proof scripts and proof terms, without conflating the two notions. The reason is that illative transformations can be seen both as proof refinement primitives when used dynamically as derivation rules, and as a compact/parallel representation of the information flow in proof objects when recorded statically inside scroll nets. This could allow for a more principled, systematic study of the properties of tactics and their interaction with proof terms, in a way that is not possible with current approaches.

References

1. Geraldine Brady and Todd H. Trimble. A categorical interpretation of C.S. Peirce's propositional logic Alpha. *Journal of Pure and Applied Algebra*, 149(3):213–239, June 2000. URL: https://linkinghub.elsevier.com/retrieve/pii/S0022404998001790, doi:10.1016/S0022-4049(98)00179-0.
2. Pablo Donato. The Flower Calculus. In Jakob Rehof, editor, *9th International Conference on Formal Structures for Computation and Deduction (FSCD 2024)*, volume 299 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:24, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSCD.2024.5.
3. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1 – 101, 1987. doi:10.1016/0304-3975(87)90045-4.
4. Jean-Yves Girard. *Proofs and Types*. Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge [England] ; New York, 1989.
5. Davide Grohmann and Marino Miculan. Directed Bigraphs. *Electronic Notes in Theoretical Computer Science*, 173:121–137, April 2007. doi:10.1016/j.entcs.2007.02.031.
6. Alessio Guglielmi. A calculus of order and interaction. Technical report, Technische Universität Dresden, 1999. URL: https://www.researchgate.net/publication/2807151_A_Calculus_of_Order_and_Interaction.
7. Willem B. Heijltjes, Dominic J. D. Hughes, and Lutz StraBburger. Intuitionistic proofs without syntax. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13, Vancouver, BC, Canada, June 2019. IEEE. URL: https://ieeexplore.ieee.org/document/8785827/, doi:10.1109/LICS.2019.8785827.
8. Willem B. Heijltjes, Dominic J. D. Hughes, and Lutz Straßburger. Normalization Without Syntax. *LIPIcs, Volume 228, FSCD 2022*, 228:19:1–19:19, 2022. doi:10.4230/LIPICS.FSCD.2022.19.
9. Dominic Hughes. Proofs without syntax. *Annals of Mathematics*, 164(3):1065–1076, 11 2006. doi:10.4007/annals.2006.164.1065.
10. Stephen Lack and Paweł Sobociński. Adhesive and quasiadhesive categories. *RAIRO - Theoretical Informatics and Applications*, 39(3):511–545, July 2005. doi:10.1051/ita:2005028.
11. Minghui Ma and Ahti-Veikko Pietarinen. A graphical deep inference system for intuitionistic logic. *Logique et Analyse*, 245:73–114, January 2019. URL: http://www.scopus.com/inward/record.url?scp=85066258215&partnerID=8YFLogxK, doi:10.2143/LEA.245.0.3285706.

[6] We only know of one attempt to give rigorous denotational semantics to tactic languages for proof refinement [25].

12 Paul-André Melliès. Functorial boxes in string diagrams. In Zoltán Ésik, editor, *Computer Science Logic*, pages 1–30, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. URL: https://www.irif.fr/~mellies/papers/Mellies06csl.pdf.

13 Dale Miller. A compact representation of proofs. *Studia Logica*, 46:347–370, 1987.

14 Robin Milner. Bigraphical Reactive Systems. In *Proceedings of the 12th International Conference on Concurrency Theory*, CONCUR '01, pages 16–35, Berlin, Heidelberg, August 2001. Springer-Verlag.

15 Leonardo de Moura and Sebastian Ullrich. The Lean 4 Theorem Prover and Programming Language. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction – CADE 28*, pages 625–635, Cham, 2021. Springer International Publishing.

16 Guillaume Munch-Maccagnoni. Focalisation and Classical Realisability. In Erich Grädel and Reinhard Kahle, editors, *Computer Science Logic*, volume 5771, pages 409–423. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. doi:10.1007/978-3-642-04027-6_30.

17 Arnold Oostra. *Los gráficos Alfa de Peirce aplicados a la lógica intuicionista*. Number 2 in Cuadernos de Sistemática Peirceana. Centro de Sistemática Peirceana, 2010. URL: https://www.academia.edu/36611719/Cuadernos_de_Sistem%C3%A1tica_Peirceana_1.

18 Michel Parigot. $\Lambda\mu$-Calculus: An algorithmic interpretation of classical natural deduction. In Andrei Voronkov, editor, *Logic Programming and Automated Reasoning*, pages 190–201, Berlin, Heidelberg, 1992. Springer. doi:10.1007/BFb0013061.

19 Charles Sanders Peirce. Prolegomena to an Apology for Pragmaticism. *The Monist*, 16(4):492–546, 1906. Publisher: Oxford University Press. URL: https://www.jstor.org/stable/27899680.

20 Ahti-Veikko Pietarinen. Peirce's game-theoretic ideas in logic. *Semiotica*, 2003:33–47, January 2003. URL: https://www.researchgate.net/publication/249934518_Peirce%27s_game-theoretic_ideas_in_logic, doi:10.1515/semi.2003.030.

21 Don D. Roberts. *The Existential Graphs of Charles S. Peirce*. De Gruyter Mouton, Berlin, Boston, 1973. doi:10.1515/9783110226225.

22 Don D. Roberts. The existential graphs. *Computers & Mathematics with Applications*, 23(6-9):639–663, March 1992. URL: https://linkinghub.elsevier.com/retrieve/pii/0898122192901274, doi:10.1016/0898-1221(92)90127-4.

23 Michele Sevegnani and Muffy Calder. Bigraphs with sharing. *Theoretical Computer Science*, 577:43–73, April 2015. doi:10.1016/j.tcs.2015.02.011.

24 Sun-Joo Shin. *The Iconic Logic of Peirce's Graphs*. The MIT Press, 05 2002. doi:10.7551/mitpress/3633.001.0001.

25 Jonathan Sterling and Robert Harper. Algebraic Foundations of Proof Refinement, March 2017. Number: arXiv:1703.05215 arXiv:1703.05215 [cs]. URL: http://arxiv.org/abs/1703.05215.

26 Lutz Straßburger. The problem of proof identity, and why computer scientists should care about hilbert's 24th problem. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 377(2140):20180038, 2019. doi:10.1098/rsta.2018.0038.

27 The Rocq Development Team. The rocq prover. Zenodo, April 2025. doi:10.5281/zenodo.15149629.

28 Andrea Aler Tubella and Lutz Straßburger. Introduction to Deep Inference. Lecture, August 2019. URL: https://inria.hal.science/hal-02390267.

OPEN+

$$\frac{v_1, \ldots, v_n \in V^+ \qquad \forall i, j \text{ such that } 1 \leq i, j \leq n \text{ and } i \neq j,\ v_i \bowtie_{\lfloor \mathfrak{S} \rfloor} v_j \qquad u, u' \notin V}{\left\langle \left\langle V \cup \{u, u'\}, \begin{array}{l} \rightarrowtriangle \setminus \{(w, v_i) \mid 1 \leq i \leq n \wedge w \rightarrowtriangle v_i\} \\ \cup \{(w, u) \mid \exists i. w \rightarrowtriangle v_i\} \\ \cup (u, u') \\ \cup \left\{(u', v_i) \;\middle|\; 1 \leq i \leq n\right\} \end{array} \right\rangle, \ell, \propto \cup (u, u') \right\rangle, \mathcal{A}, \langle \longleftrightarrow \cup (u, u'), \mathrel{-\!\!\times\!\!-} \rangle}$$

OPEN−

$$\frac{v_1, \ldots, v_n \in V^- \qquad \forall i, j \text{ such that } 1 \leq i, j \leq n \text{ and } i \neq j,\ v_i \bowtie_{\lfloor \mathfrak{S} \rfloor} v_j \qquad u, u' \notin V}{\left\langle \left\langle V \cup \{u, u'\}, \begin{array}{l} \rightarrowtriangle \setminus \{(w, v_i) \mid 1 \leq i \leq n \wedge w \rightarrowtriangle v_i\} \\ \cup \{(w, u) \mid \exists i. w \rightarrowtriangle v_i\} \\ \cup (u, u') \\ \cup \left\{(u', v_i) \;\middle|\; 1 \leq i \leq n\right\} \end{array} \right\rangle, \ell, \propto \cup (u, u') \right\rangle, \mathcal{A}, \langle \longleftrightarrow, \mathrel{-\!\!\times\!\!-} \cup (u, u') \rangle}$$

CLOSE+

$$\frac{v \in V^+ \qquad v \propto u \qquad (v, u) \notin \mathrel{-\!\!\times\!\!-}}{\Phi, \mathcal{A}, (\longleftrightarrow, \mathrel{-\!\!\times\!\!-} \cup (v, u))}$$

CLOSE−

$$\frac{v \in V^- \qquad v \propto u \qquad (v, u) \notin \longleftrightarrow}{\Phi, \mathcal{A}, (\longleftrightarrow \cup (v, u), \mathrel{-\!\!\times\!\!-})}$$

INSERT

$$\frac{v \in V^+ \setminus \mathrm{dom}(\ell) \qquad \phi \text{ is a scroll structure with a single root } u}{\langle\langle V \uplus V_\phi, (\rightarrowtriangle \uplus \rightarrowtriangle_\phi) \cup (v^\triangleleft, u^\triangleright)\rangle, \ell \uplus \ell_\phi, \propto \uplus \propto_\phi\rangle, (\curvearrowright, \circlearrowright \cup u^\triangleright), \mathcal{I}}$$

DELETE

$$\frac{v \in V^+ \qquad v \notin \circlearrowright}{\Phi, \langle \curvearrowright, \circlearrowright \cup v \rangle, \mathcal{I}}$$

ITERATEROOT

$$\frac{\rightrightarrows u = \varnothing}{\left\langle \left\langle V \uplus V_{\lfloor u \downarrow \rfloor}, \rightarrowtriangle \uplus \rightarrowtriangle_{\lfloor u \downarrow \rfloor} \right\rangle, \ell \uplus \ell_{\lfloor u \downarrow \rfloor}, \propto \uplus \propto_{\lfloor u \downarrow \rfloor} \right\rangle, \langle \curvearrowright \cup (u^\triangleleft, u^\triangleright), \circlearrowright \rangle, \mathcal{I}}$$

ITERATEDEEP

$$\frac{v \in V^- \setminus \mathrm{dom}(\ell) \qquad \neg \exists u_0.\, u_0 \propto u \qquad \exists v_0.\, u \bowtie v_0 \wedge v_0 \rightarrowtriangle^* v}{\left\langle \left\langle V \uplus V_{\lfloor u \downarrow \rfloor}, \rightarrowtriangle \uplus \rightarrowtriangle_{\lfloor u \downarrow \rfloor} \cup (v^\triangleleft, u^\triangleright) \right\rangle, \ell \uplus \ell_{\lfloor u \downarrow \rfloor}, \propto \uplus \propto_{\lfloor u \downarrow \rfloor} \right\rangle, \langle \curvearrowright \cup (u^\triangleleft, u^\triangleright), \circlearrowright \rangle, \mathcal{I}}$$

DEITERATE

$$\frac{v \in V^- \qquad \neg \exists u_0.\, u_0 \propto u \qquad \exists v_0.\, u \bowtie v_0 \wedge v_0 \rightarrowtriangle^* v \qquad \lfloor u \downarrow \rfloor \simeq \lceil v \downarrow \rceil}{\Phi, \langle \curvearrowright \cup (u, v), \circlearrowright \rangle, \mathcal{I}}$$

Figure 7 Derivation rules for scroll nets

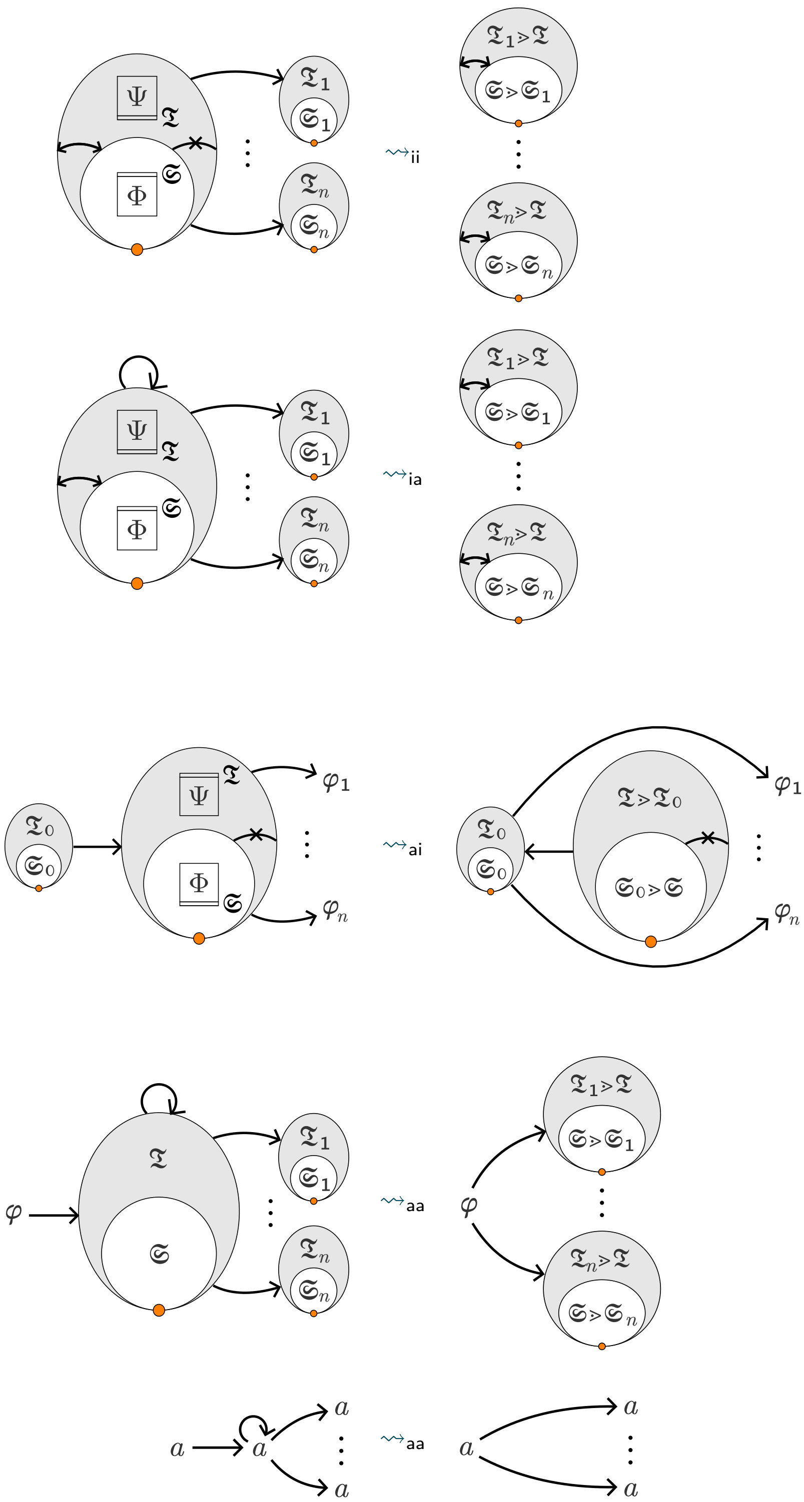


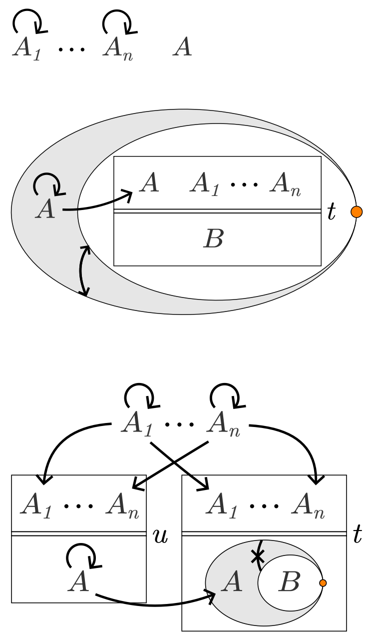
Figure 8 Detour reduction rules

Let $\Gamma = A_1, \ldots, A_n$.

$$\frac{}{\Gamma, x : A \vdash x : A}\,\mathsf{var} \quad \mapsto$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\,\mathsf{abs} \quad \mapsto$$

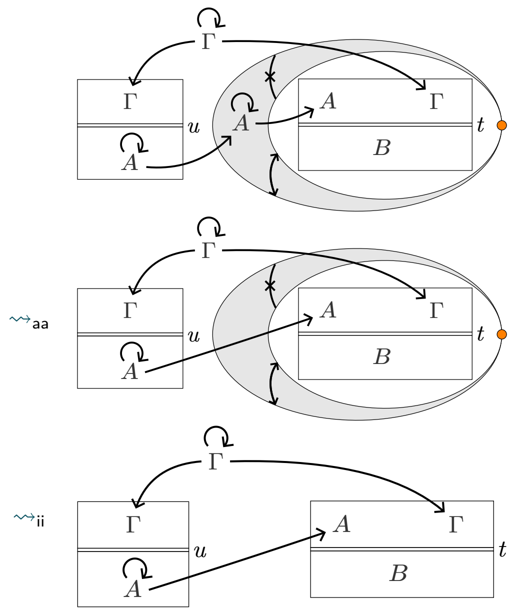
$$\frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash (t)u : B}\,\mathsf{app} \quad \mapsto$$



(a) Typing rules

$$\cfrac{\cfrac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\,\mathsf{abs} \qquad \Gamma \vdash u : A}{\Gamma \vdash (\lambda x.t)u : B}\,\mathsf{app}$$

$\rightsquigarrow_\beta \quad \Gamma \vdash t\{u/x\} : B$



(b) β-reduction

Figure 9 Simulation of simply typed λ-calculus