First-Order Intuitionistic Linear Logic and Hypergraph Languages

Tikhon Pshenitsyn ✉ ⌂ iD
Steklov Mathematical Institute of RAS, 8 Gubkina St., Moscow 119991, Russian Federation

—— **Abstract** ——————————————————————————

The Lambek calculus is a substructural logic known to be closely related to the formal language theory: on the one hand, it is used for generating formal languages by means of categorial grammars and, on the other hand, it has formal language semantics, with respect to which it is sound and complete. This paper studies a similar relation between first-order intuitionistic linear logic ILL1 along with its multiplicative fragment MILL1 on the one hand and the hypergraph grammar theory on the other. In the first part, we introduce a novel concept of hypergraph first-order logic categorial grammar, which is a generalisation of string MILL1 grammars studied e.g. in Richard Moot's 2014 works. We prove that hypergraph ILL1 grammars generate all recursively enumerable hypergraph languages and that hypergraph MILL1 grammars are as powerful as linear-time hypergraph transformation systems. In addition, we show that the class of languages generated by string MILL1 grammars is closed under intersection and that it includes a non-semilinear language as well as an NP-complete one. This shows how much more powerful string MILL1 grammars are as compared to Lambek categorial grammars.

In the second part, we develop hypergraph language models for MILL1. In such models, formulae of the logic are interpreted as hypergraph languages and multiplicative conjunction is interpreted using parallel composition, which is one of the operations of HR-algebras introduced by Courcelle. We prove completeness of the universal-implicative fragment of MILL1 with respect to these models and thus present a new kind of semantics for a fragment of first-order linear logic.
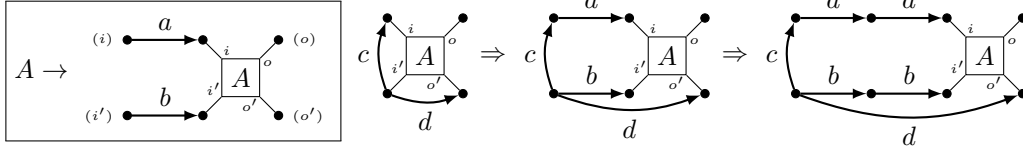
1 Introduction

There is a strong connection between substructural logics, especially non-commutative ones, and the theory of formal languages and grammars [5, 23]. This connection is two-way. On the one hand, a logic can be used as a derivational mechanism for generating formal languages, which is the essence of categorial grammars. One prominent example is Lambek categorial grammars based on the Lambek calculus L [18]; formulae of the latter are built using multiplicative conjunction '·' and two directed implications '\', '/'. In a Lambek categorial grammar, one assigns a finite number of formulae of L to each symbol of an alphabet and chooses a distinguished formula S; then, the grammar accepts a string $a_1 \ldots a_n$ if the sequent $A_1, \ldots, A_n \vdash S$ is derivable in L where, for $i = 1, \ldots, n$, A_i is one of the formulae assigned to a_i. A famous result by Pentus [25] says that Lambek categorial grammars generate exactly context-free languages (without the empty word, to be precise).

On the other hand, algebras of formal languages can serve as models for substructural logics. For example, one can define language semantics for the Lambek calculus as follows:

a language model is a function u mapping formulas of L to formal languages such that $u(A \cdot B) = \{vw \mid v \in u(A), w \in u(B)\}$, $u(B \backslash A) = \{w \mid \forall v \in u(B)\ vw \in u(A)\}$, and $u(A/B) = \{v \mid \forall w \in u(B)\ vw \in u(A)\}$; a sequent $A \vdash B$ is interpreted as inclusion $u(A) \subseteq u(B)$. Another famous result by Pentus [26] is that L is sound and complete w.r.t. language semantics; strong completeness for the fragment of L without '·' had been proved earlier by Buszkowski in [4] using canonical models.

Numerous variants and extensions of the Lambek calculus have been studied, including its nonassociative version [23], commutative version [36] (i.e. multiplicative intuitionistic linear logic), the multimodal Lambek calculus [19], the displacement calculus [24] etc. These logics have many common properties, which motivates searching for a unifying logic. One such "umbrella" logic is the first-order multiplicative intuitionistic linear logic MILL1 [20, 22], which is the multiplicative fragment of first-order intuitionistic linear logic ILL1. The Lambek calculus can be embedded in MILL1 [22]: for example, the L sequent $p \cdot p \backslash q \vdash q$ is translated into the MILL1 sequent $p(x_0, x_1) \otimes \forall y(p(y, x_1) \multimap q(y, x_2)) \vdash q(x_0, x_2)$. In such a translation, variables "fix" the order of formulae. Although MILL1 is a first-order generalisation of L, derivability problems in these logics have the same complexity, namely, they are NP-complete. One can define MILL1 categorial grammars in the same manner as Lambek categorial grammars [20, 33]. The former generalise the latter, hence generate all context-free languages (see the definition of the latter in [14]). Moot proved in [20] that MILL1 grammars generate all multiple context-free languages, hence some non-context-free ones, e.g. $\{ww \mid w \in \Sigma^*\}$.

It turns out that the interplay between propositional substructural logics and formal grammars can be elevated fruitfully to that between first-order substructural logics and hypergraph grammars. This is the subject of this article. Hypergraph grammar approaches generate sets of hypergraphs; usually they are designed as generalisations of grammar formalisms for strings. For example, hyperedge replacement grammar [8] is a formalism that extends context-free grammar. A rule of a hyperedge replacement grammar allows one to replace a hyperedge in a hypergraph by another hypergraph; see an example below.

◼   **Figure 1** A hyperedge replacement rule (in a box) and an example of it being applied twice.

Note that, in this approach, hyperedges but not nodes are labeled. Some of the nodes, called external, are distinguished in a hypergraph; e.g., in the above example, these are the nodes marked by $(i), (o), (i'), (o')$. External nodes are needed to specify how hyperedge replacement is done. A more general approach, which corresponds to type-0 grammars in the Chomsky hierarchy, is hypergraph transformation systems (the term is taken from [17]), which allow one to replace a subhypergraph in a hypergraph by another hypergraph.

Naturally, a hypergraph can be represented by a linear logic formula. Namely, one can interpret hyperedges as predicates, nodes as variables, and external nodes as free variables. For example, the hypergraph in the box from Figure 1 can be converted into the formula $\exists x. \exists y. a(i, x) \otimes b(i', y) \otimes A(x, o, y, o')$. This idea underlies the concept of *hypergraph first-order categorial grammars*, which we introduce in Section 3. Roughly speaking, given a first-order logic $\mathcal{L}$, say, MILL1, a hypergraph $\mathcal{L}$ categorial grammar takes a hypergraph, assigns a formula of $\mathcal{L}$ to each its hyperedge and node, converts the resulting hypergraph into a $\mathcal{L}$

sequent and check whether it is derivable in $\mathcal{L}$. Using first-order linear logic for generating hypergraph languages is a novel idea which has not yet been explored in the literature. In Section 3, we study expressive power of grammars defined thusly and prove the following.

1. *Hypergraph* ILL1 *grammars are equivalent to hypergraph transformation systems and thus they generate all recursively enumerable hypergraph languages (Theorem 23).* This result relates hypergraph ILL1 grammars to the well studied approach in the field of hypergraph grammars based on the double pushout graph transformation procedure.

2. *Hypergraph* MILL1 *grammars are at least as powerful as linear-time hypergraph transformation systems (Theorem 25).* The latter are hypergraph transformation systems where the length of a derivation is bounded by a linear function w.r.t. the size of the resulting hypergraph. The linear-time bound has been studied for many grammar formalisms [2, 11, 29, 34], but, to our best knowledge, it is the first time it is used for graph grammars.

The proofs partially use the techniques from [29] where languages generated by grammars over the commutative Lambek calculus are studied. As compared to [29], the proofs in this paper are more technically involved because of complications arising when working with quantifiers and variables in the first-order setting.

In Section 4, using the methods developed for hypergraph MILL1 grammars, we discover the following properties of the class of languages generated by string MILL1 grammars.

1. *Languages generated by string* MILL1 *grammars are closed under intersection (Theorem 26). Consequently, non-semilinear languages, e.g. the language* $\{a^{2n^2} \mid n \in \mathbb{N}\}$*, can be generated by string* MILL1 *grammars.*

2. *String* MILL1 *grammars generate an NP-complete language (Theorem 30).* Note that the Lambek calculus is NP-complete [27] but Lambek categorial grammars generate only context-free languages, which are in P. The logic MILL1 is NP-complete as well but string MILL1 grammars are able to generate non-polynomial languages (assuming P$\neq$NP), hence they are much more powerful.

The question whether string MILL1 grammars generate only multiple context-free grammars was left open by Richard Moot in [20], and I considered Theorem 30 to be the first one answering it. Recently, however, Sergei Slavnov pointed out to an alternative answer to this question, using previously known techniques. Namely, in [21], it is shown that hybrid type-logical grammars can be translated into MILL1; hybrid type-logical grammars generalise abstract categorial grammars, and it is proved in [31] that the latter generate an NP-complete language. Our proof relies on a different technique, namely, on reducing linear-time hypergraph transformation systems, which are a rule-based approach unlike hybrid type-logical grammars. In general, finding a natural rule-based formalism equivalent to a MILL1 grammars is an interesting question to study, and Theorem 30 is a step towards the answer[1].

The second part of the paper (Section 5), not related to the first one, is devoted to developing hypergraph language semantics for MILL1, thus establishing the other way round connection between first-order linear logic and hypergraph languages. Linear logic is considered as a logic for reasoning about resources [10], and language models for the Lambek calculus are one of formalisations of this statement with resources being words; this agrees with linguistic applications of L. In this paper, we shall show that hypergraphs can be treated as "first-order resources." In a hypergraph language model (Definition 33), MILL1 formulas

---

[1] We believe that linear-time hypergraph transformation systems are essentially equivalent to MILL1 grammars in terms of generative power because, as we conjecture, it is possible to simulate MILL1 axioms and rules by hypergraph transformation rules and hence to convert hypergraph MILL1 grammars into linear-time transformation systems. However, some subtleties arise related to the definition of hypergraph transformation rules when one tries to do so; we discuss them after the proof of Theorem 25.

are interpreted by sets of hypergraphs, and the tensor operation $A \otimes B$ is interpreted using the parallel composition operation. The latter one is "gluing" of hypergraphs; it is studied well in the hypergraph grammar theory, namely, it is one of the operations in the language of HR-algebras [7]. Hypergraph language models is a particular case of intuitionistic phase semantics (see the definition in [15]) with the trivial closure operator $\mathrm{Cl}(X) = X$.

Our main result concerning hypergraph language models is soundness of MILL1 and completeness of its $\{\multimap, \forall\}$-fragment w.r.t. them (Theorem 35). The proof is inspired by Buszkowski's one [4] but it is more technically involved, again because of the first-order setting. This result's importance amounts to the fact that hypergraph language models is one of few, to our best knowledge, examples of a specific semantics for a fragment of first-order intuitionistic linear logic, which, moreover, is grounded in the hypergraph language theory.

2   Preliminaries

In Section 2.1, we introduce notions from the field of graph grammars, and, in Section 2.2, we introduce first-order intuitionistic linear logic.

2.1   Hypergraphs & Hypergraph Transformation Systems

There are many paradigms in the field of graph grammars, including node replacement grammars, hyperedge replacement grammars, algebraic approaches (double pushout, single pushout) with a more categori**c**al flavour, definability in monadic second-order logic etc. We shall work with the definition of a hypergraph from the field of hyperedge replacement grammars [8, 9, 12] because it fits first-order linear logic better than other ones. In hypergraphs we shall deal with, only hyperedges are labeled while nodes play an auxiliary role. Some of the nodes are marked as external; informally, they play the role of gluing points. Throughout the paper, we shall explore a natural correspondence between hypergraphs defined thusly and first-order linear logic formulae. In contrast, there would be no such correspondence if we sticked to the definition of a hypergraph where nodes are labeled and edges are not.

We fix a countable set Σ; its elements are called *selectors*. In the grammar-logic correspondence we shall develop, selectors will guide variable substitution.

▶ **Definition 1.** *A Σ-typed alphabet is a set C along with a function* $\mathrm{type} : C \to \mathcal{P}(\Sigma)$ *such that* $\mathrm{type}(c)$ *is finite for $c \in C$.*
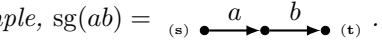
▶ **Definition 2.** *Let C be a finite Σ-typed alphabet of hyperedge labels. A* hypergraph *over C is a tuple $H = \langle V_H, E_H, lab_H, att_H, ext_H \rangle$ where V_H is a finite set of nodes; E_H is a finite set of hyperedges; for each $e \in E_H$, $lab_H : E_H \to C$ is the labeling function and $att_H(e) : \mathrm{type}(lab_H(e)) \to V_H$ is the attachment function; $ext_H : \mathrm{type}(H) \to V_H$ is a function with the domain $\mathrm{type}(H) \subseteq \Sigma$. Elements of $\mathrm{ran}(ext_H)$ are called* external nodes. *The set of hypergraphs over C is denoted by $\mathcal{H}(C)$. Let $\mathrm{type}_H(e) := \mathrm{dom}(att_H(e))$ for each $e \in E_H$.*

In drawings of hypergraphs, nodes are depicted as black circles and hyperedges are depicted as labeled rectangles. When depicting a hypergraph H, we draw a line with a label σ from e to v if $att_H(e)(\sigma) = v$. External nodes are represented by symbols in round brackets: if $ext_H(\sigma) = v$, then we mark v as (σ).
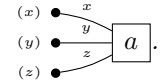
There is a standard issue with distinguishing between concrete and abstract hypergraphs, i.e. between hypergraphs and their isomorphism classes. When one considers a hypergraph language L, it is reasonable to assume that it consists of abstract hypergraphs (or, equivalently, that it is closed under isomorphism); however, when we write $H \in L$, we assume that H is a

concrete hypergraph. Following tradition, we often do not distinguish between abstract and concrete hypergraphs to avoid excessive bureaucracy.

Let us fix two selectors, $\mathbf{s}$ and $\mathbf{t}$. If $\mathrm{type}_H(e) = \{\mathbf{s}, \mathbf{t}\}$, then the hyperedge e is called an *edge* and it is depicted by an arrow going from $att_H(e)(\mathbf{s})$ to $att_H(e)(\mathbf{t})$.

▶ **Definition 3.** A string graph $\mathrm{sg}(w)$ induced by a string $w = a_1 \ldots a_n$ is defined as follows: $V_{\mathrm{sg}(w)} = \{v_0, \ldots, v_n\}$, $E_{\mathrm{sg}(w)} = \{e_1, \ldots, e_n\}$; $\mathrm{type}(e_i) = \mathrm{type}(\mathrm{sg}(w)) = \{\mathbf{s}, \mathbf{t}\}$, $att_{\mathrm{sg}(w)}(e_i)(\mathbf{s}) = v_{i-1}$, $att_{\mathrm{sg}(w)}(e_i)(\mathbf{t}) = v_i$, $lab_{\mathrm{sg}(w)}(e_i) = a_i$ (for $i = 1, \ldots, n$); $ext_{\mathrm{sg}(w)}(\mathbf{s}) = v_0$, $ext_{\mathrm{sg}(w)}(\mathbf{t}) = v_n$. For example, $\mathrm{sg}(ab) = {}_{(\mathbf{s})} \bullet \xrightarrow{a} \bullet \xrightarrow{b} \bullet_{(\mathbf{t})}$.

▶ **Definition 4.** Given $a \in C$, $a^\bullet$ is a hypergraph such that $V_{a^\bullet} = \mathrm{type}(a)$; $E_{a^\bullet} = \{e\}$ with $\mathrm{type}_{a^\bullet}(e) = \mathrm{type}(a)$; $att_{a^\bullet}(\sigma) = ext_{a^\bullet}(\sigma) = \sigma$ for $\sigma \in \mathrm{type}(a)$. For example, if $\mathrm{type}(a) = \{x, y, z\}$, then $a^\bullet = $.

▶ **Definition 5.** If H_1, H_2 are hypergraphs with $\mathrm{type}(H_1) \cap \mathrm{type}(H_2) = \emptyset$, then their disjoint union *is the hypergraph* $H_1 + H_2 := \langle V_{H_1} \sqcup V_{H_2}, E_{H_1} \sqcup E_{H_2}, att, lab, ext \rangle$ where $att = att_{H_i}$, $lab = lab_{H_i}$ on E_i for $i = 1, 2$, and ext is the union of functions ext_{H_1} and ext_{H_2}.

Now, let us define the hyperedge replacement operation.

▶ **Definition 6.** Let H be a hypergraph and let R be a binary relation on V_H. Let $\equiv_R$ be the smallest equivalence relation on V_H containing R. Then $H/R = H'$ is the following hypergraph: $V_{H'} = \{[v]_{\equiv_R} \mid v \in V_H\}$; $E_{H'} = E_H$; $lab_{H'} = lab_H$; $att_{H'}(e)(s) = [att_H(e)(s)]_{\equiv_R}$; $ext_{H'}(s) = [ext_H(s)]_{\equiv_R}$.
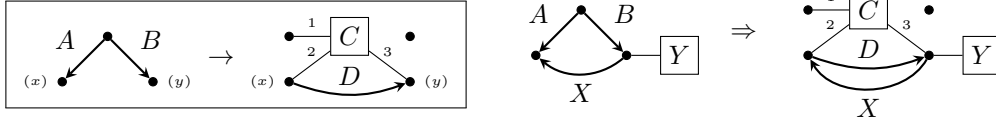
▶ **Definition 7.** Let H, K be two hypergraphs over C; let $e \in E_H$ be a hyperedge such that $\mathrm{type}(e) = \mathrm{type}(K)$. Then the replacement of e by K in H (the result being denoted by $H[e/K]$) is defined as follows:
1. *Remove e from H and add a disjoint copy of K. Formally, let L be the hypergraph such that $V_L = V_H \sqcup V_K$, $E_L = (E_H \setminus \{e\}) \sqcup E_K$, lab_L is the restriction of $lab_H \cup lab_K$ to E_L, att_L is the restriction of $att_H \cup att_K$ to E_L, and $ext_L = ext_H$.*
2. *Glue the nodes that are incident to e in H with the external nodes of K. Namely, let $H[e/K] := L/R$ where $R = \{(att_H(e)(s), ext_K(s)) \mid s \in \mathrm{type}(e)\}$.*

Using hyperedge replacement, we define *hypergraph transformation system*, a formalism which shall be used to describe expressive power of hypergraph categorial grammars. It enables one to replace a subhypergraph in a hypergraph with another hypergraph.

▶ **Definition 8.** A hypergraph transformation rule (ht-rule) *is of the form* $r = (H \to H')$ *where H, H' are hypergraphs such that* $\mathrm{type}(H) = \mathrm{type}(H')$ *and* ext_H, $ext_{H'}$ *are injective. We say that G is transformed into G' via r and denote this by $G \Rightarrow_r G'$ (also by $G \Rightarrow G'$) if $G = K[e/H]$ and $G' = K[e/H']$ for some K and $e \in E_K$ such that $att_K(e)$ is injective. A hypergraph transformation system (ht-system) is a tuple $\mathcal{G} = \langle N, T, P, S \rangle$ where N, T are Σ-typed alphabets, P is a finite set of hypergraph transformation rules and $S \in \mathcal{H}(N \cup T)$ is a start hypergraph such that ext_S is injective. The language $L(\mathcal{G})$ generated by $\mathcal{G}$ consists of hypergraphs $H \in \mathcal{H}(T)$ such that $S \Rightarrow^* H$.*

The definition of a hypergraph transformation rule, although given in a slightly unconventional way through hyperedge replacement, coincides with the standard notion of a graph transformation rule with injective morphisms of the double pushout approach formalism

Figure 2 An example of a hypergraph transformation rule (boxed) and of its application.

in the corresponding category of hypergraphs; compare it with [17]. Also, our definition is essentially the same as that from [35] (with the only difference that the cited paper deals with graphs rather than with hypergraphs). In that paper, the following proposition is proved.

▶ **Proposition 9.** *Hypergraph transformation systems generate all recursively enumerable hypergraph languages.*

This is a rather expected result related to string rewriting systems generating all recursively enumerable string languages. To prove Proposition 9, it suffices to show how to convert a string representation of a hypergraph into the hypergraph itself by means of ht-rules.

The injectivity requirements in Definition 8 are quite standard, ht-systems defined thusly correspond to the class of $\mathrm{DPO}^{i/i}$ grammars investigated in [13] ("DPO" stands for "double-pushout approach"). In this paper, injectivity is crucial in the proof of Theorem 25.

2.2 First-Order Intuitionistic Linear Logic

We assume the reader's familiarity with the basic principles and issues of first-order logic. Let us fix a countable set of variables Var and a countable set of predicate symbols with arities. Atomic formulae are of the form $p(x_1, \ldots, x_n)$ where p is a predicate symbol of arity n and $x_1, \ldots, x_n$ are variables. Following [16, 20, 22], we do not allow function symbols; note that complex terms would not fit in the *variables are nodes, predicates are hyperedges* paradigm. We also do not allow constants because they can easily be simulated by variables.

Formulae of intuitionistic linear logic ILL1 are built from atomic formulae and propositional constants $0, 1, \top$ using the multiplicative connectives $\otimes, \multimap$, the additive ones $\wedge, \vee$, and the exponential one ! along with the quantifiers $\exists, \forall$. The multiplicative fragment of ILL1 denoted by MILL1 does not have constants and uses only $\otimes, \multimap$ and $\exists, \forall$. A sequent is a structure of the form $\Gamma \vdash B$ where Γ is a multiset of formulae and B is a formula.

Note that we shall sometimes describe multisets using the notation $\{f(x) \mid \Phi(x)\}$. An element a belongs to this multiset n times if there are exactly n elements $x_1, \ldots, x_n$ such that $f(x_i) = a$ and such that x_i satisfies Φ.

The only axiom is $A \vdash A$. The rules for MILL1 are presented below.

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C} \ (\otimes L) \quad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \ (\otimes R) \quad \frac{\Pi \vdash B \quad \Gamma, A \vdash C}{\Gamma, \Pi, B \multimap A \vdash C} \ (\multimap L) \quad \frac{\Gamma, B \vdash A}{\Gamma \vdash B \multimap A} \ (\multimap R)$$

$$\frac{\Gamma, A[z/x] \vdash B}{\Gamma, \exists x A \vdash B} \ (\exists L) \qquad \frac{\Gamma \vdash A[y/x]}{\Gamma \vdash \exists x A} \ (\exists R) \qquad \frac{\Gamma, A[y/x] \vdash B}{\Gamma, \forall x A \vdash B} \ (\forall L) \qquad \frac{\Gamma \vdash A[z/x]}{\Gamma \vdash \forall x A} \ (\forall R)$$

Here y is any variable while z is a variable which is not free in Γ, A, B; $A[y/x]$ denotes replacing all free occurrences of x in A by y. More generally, if $h : \mathrm{FVar}(A) \to \mathrm{Var}$ is a function defining a correct substitution, then $A[h]$ denotes the result of substituting $h(x)$ for x for $x \in \mathrm{FVar}(A)$ ($\mathrm{FVar}(A)$ is the set of free variables in A). Rules for additive and exponential connectives of ILL1 can be found e.g. in [32, Appendix E].

The cut rule is admissible in ILL1:

$$\frac{\Gamma \vdash A \quad A, \Delta \vdash C}{\Gamma, \Delta \vdash C} \ (\mathrm{cut})$$

Using it, one can prove that the rules $(\otimes L)$, $(\multimap R)$, $(\exists L)$, and $(\forall R)$ are invertible, i.e. that, if a conclusion of any of these rules is provable in ILL1, then so is its premise.

3 Hypergraph First-Order Categorial Grammars

The idea of extending the Lambek calculus and categorial grammars to hypergraphs was explored recently in the work [28], where the hypergraph Lambek calculus was introduced. This is a propositional logic whose formulae are built using two operators, $\times$ and $\div$ (similar to linear logic $\otimes$ and $\multimap$). Formulae of this calculus can be used as labels on hyperedges: e.g. if H is a hypergraph labeled by formulas, then $\times(H)$ is a formula. Based on the hypergraph Lambek calculus, hypergraph Lambek grammars were defined and their properties were investigated. Although the definition of the hypergraph Lambek calculus is justified in [28], the syntax of this logic is somewhat cumbersome. We are going to show that one can use any first-order logic, such as MILL1, as the underlying logic for hypergraph categorial grammars. The definitions we propose below are much simpler than those from [28]; besides, they enable one to rely on the well studied apparatus of linear logic.

Let us start with the definition of a string categorial grammar over a first-order logic. This notion appears in [20, 33] for MILL1 but we would like to start with a more general exposition. Let $\mathcal{L}$ be a first-order sequent calculus of interest and let $\mathrm{Fm}(\mathcal{L})$ denote the set of its formulas. Let $\mathbf{s}, \mathbf{t}$ be two fixed variables (note that earlier we used them as selectors).

▶ **Definition 10.** *A string $\mathcal{L}$ grammar is a tuple $\mathcal{G} = \langle T, S, \rhd \rangle$ where T is a finite alphabet, S is a formula of $\mathcal{L}$ such that $\mathrm{FVar}(S) \subseteq \{\mathbf{s}, \mathbf{t}\}$, and $\rhd \subseteq T \times \mathrm{Fm}(\mathcal{L})$ is a finite binary relation such that $a \rhd A$ implies $\mathrm{FVar}(A) \subseteq \{\mathbf{s}, \mathbf{t}\}$. The language $L(\mathcal{G})$ generated by $\mathcal{G}$ is defined as follows: $a_1 \ldots a_n \in L(\mathcal{G})$ if and only if there are formulas $A_1, \ldots, A_n$ such that $a_i \rhd A_i$ for $i = 1, \ldots, n$ and such that the sequent $A_1[x_0/\mathbf{s}, x_1/\mathbf{t}], \ldots, A_n[x_{n-1}/\mathbf{s}, x_n/\mathbf{t}] \vdash S[x_0/\mathbf{s}, x_n/\mathbf{t}]$ is derivable in $\mathcal{L}$ where $x_0, \ldots, x_n$ are distinct variables.*

▶ **Example 11.** Let $T = \{a\}$, let $S = q(\mathbf{s}, \mathbf{t})$, and let $\rhd$ consist of the pairs $a \rhd p(\mathbf{s}, \mathbf{t})$, $a \rhd \forall x.p(x, \mathbf{s}) \multimap q(x, \mathbf{t})$. This grammar accepts the string aa, because the sequent

$$p(x_0, x_1), \forall x.p(x, x_1) \multimap q(x, x_2) \vdash q(x_0, x_2)$$

is derivable in MILL1.

We see that, in Definition 10, the noncommutative structure of a string is simulated by variables. Informally, one could imagine a string graph with the nodes $x_0, \ldots, x_n$ such that, for $i = 1, \ldots, n$, there is an edge labeled by A_i connecting x_{i-1} to x_i. Based on this observation, let us introduce the central notion of hypergraph $\mathcal{L}$ grammars. From now on, we consider nodes, selectors and logical variables as objects of the same kind. Besides, if T is a Σ-typed alphabet, then we treat $a \in T$ as a predicate symbol.

▶ **Definition 12.** *Let us fix a variable $x_\bullet$ and a symbol $\bullet$. A hypergraph $\mathcal{L}$ grammar is a quadruple $\mathcal{G} = \langle T, S, X, \rhd \rangle$ where T is a Σ-typed alphabet; $\rhd \subseteq (T \cup \{\bullet\}) \times \mathrm{Fm}(\mathcal{L})$ is a finite binary relation such that $a \rhd A$ implies $\mathrm{FVar}(A) \subseteq \mathrm{type}(a)$ and $\bullet \rhd A$ implies $\mathrm{FVar}(A) \subseteq \{x_\bullet\}$; finally, S is a formula of $\mathcal{L}$ such that $\mathrm{FVar}(S) \subseteq X \subseteq \Sigma$.*

▶ **Definition 13.** *The language $L(\mathcal{G})$ is defined as follows: $H \in L(\mathcal{G})$ if and only if $\text{type}(H) = X$ and there are functions $h_V : V_H \to \text{Fm}(\mathcal{L})$, $h_E : E_H \to \text{Fm}(\mathcal{L})$ such that*

1. *$\bullet \triangleright h_V(v)$ for $v \in V_H$, $lab_H(e) \triangleright h_E(e)$ for $e \in E_H$;*

2. *the sequent $\{h_E(e)[att_H(e)] \mid e \in E_H\}, \{h_V(v)[v/x_\bullet] \mid v \in V_H\} \vdash S[ext_H]$ is derivable in $\mathcal{L}$.*

▶ **Example 14.** Let $\mathcal{G}$ be a hypergraph MILL1 grammar with $T := \{a, b\}$ ($\text{type}(a) = \{\mathbf{s}, \mathbf{t}\}$, $\text{type}(b) = \{1, 2, 3\}$), $X := \{\mathbf{s}, \mathbf{t}\}$, $S := p(\mathbf{s}, \mathbf{t}) \otimes r \otimes r \otimes r$, and with $\triangleright$ consisting of the pairs

- $a \triangleright q(\mathbf{s}, \mathbf{t})$; $a \triangleright \forall x.q(x, \mathbf{s}) \multimap p(x, \mathbf{t})$;
- $b \triangleright q(2, 3) \multimap p(1, 1) \multimap p(2, 3)$;
- $\bullet \triangleright r$; $\bullet \triangleright \forall y.p(x_\bullet, y)$.

Consider the hypergraph $H =$ with $V_H = \{v_1, v_2, v_3, v_4\}$ (nodes are enumerated left to right, top to bottom). It belongs to $L(\mathcal{G})$, because the sequent

$$q(v_3, v_4), q(v_3, v_4) \multimap p(v_1, v_1) \multimap p(v_3, v_4), \forall y.p(v_1, y), r, r, r \vdash p(v_3, v_4) \otimes r \otimes r \otimes r$$

is derivable in MILL1. In this sequent, the first formula corresponds to the a-labeled hyperedge, the second one corresponds to the b-labeled hyperedge, the third formula corresponds to v_1 and the remaining formulae correspond to v_2, v_3, v_4.

The string graph $\text{sg}(aa) =$ also belongs to $L(\mathcal{G})$, because the sequent

$$q(v_0, v_1), \forall x.q(x, v_1) \multimap p(x, v_2), r, r, r \vdash p(v_0, v_2) \otimes r \otimes r \otimes r$$

is derivable in MILL1 (the nodes of $\text{sg}(aa)$ from left to right are v_0, v_1, v_2).

Finally, the hypergraph (s) $\bullet$ $\bullet$ $\bullet$ $\bullet$ (t) without hyperedges and with nodes w_1, w_2, w_3, w_4 is also accepted by $\mathcal{G}$, because the following sequent is derivable in MILL1:

$$\forall y.p(w_1, y), r, r, r \vdash p(w_1, w_4) \otimes r \otimes r \otimes r.$$

Let us comment on Definitions 12 and 13. The relation $\triangleright$ assigns formulae of $\mathcal{L}$ to hyperedge labels from T. Besides, it assigns formulae with the free variable $x_\bullet$ (or without free variables) to the distinguished "node symbol" $\bullet$. Since, in the theory of hyperedge replacement, it is traditional to consider hypergraphs where only hyperedges are labeled, one might ask why we assign formulas not only to hyperedges but to nodes as well. The answer is that we need to have some control over nodes. If we remove all the parts concerning nodes from Definitions 12 and 13, then hypergraph $\mathcal{L}$ grammars would completely ignore isolated nodes; this is a minor yet annoying issue. Moreover, each hypergraph language generated by a hypergraph $\mathcal{L}$ grammar (for, say, $\mathcal{L} = \text{MILL1}$) would be closed under node identification.

▶ **Example 15.** Assume that $\text{sg}(aa) \in L(\mathcal{G})$ for a hypergraph MILL1 grammar $\mathcal{G} = \langle T, S, \{\mathbf{s}, \mathbf{t}\}, \triangleright \rangle$ where nodes do not participate in the grammar formalism. This would mean that there are formulae A_1, A_2 with free variables in $\{\mathbf{s}, \mathbf{t}\}$ such that $a \triangleright A_1$, $a \triangleright A_2$ and such that the sequent $A_1[v_0/\mathbf{s}, v_1/\mathbf{t}], A_2[v_1/\mathbf{s}, v_2/\mathbf{t}] \vdash S[v_0/\mathbf{s}, v_2/\mathbf{t}]$ is derivable in MILL1. However, this necessarily implies that the sequent $A_1[v_0/\mathbf{s}, v_0/\mathbf{t}], A_2[v_0/\mathbf{s}, v_1/\mathbf{t}] \vdash S[v_0/\mathbf{s}, v_1/\mathbf{t}]$ is derivable in MILL1 too, hence the hypergraph is also accepted by $\mathcal{G}$. Consequently, hypergraph MILL1 grammars without node-formula assignment are not able to generate a language consisting only of string graphs, which is quite undesirable.

Another remark concerning Definition 12 is why we need the set X. This set makes the language generated by a grammar consistent in terms of external nodes: if $H_1, H_2 \in L(\mathcal{G})$, then $\text{type}(H_1) = \text{type}(H_2) = X$. Note that ht-systems and hyperedge replacement grammars [8] are consistent in this sense.

Given a hypergraph $\mathcal{L}$ grammar $\mathcal{G}$, one can consider only string graphs generated by it and thus associate a string language with Γ.

▶ **Definition 16.** *The string language $L^{str}(\mathcal{G})$ generated by a hypergraph $\mathcal{L}$ grammar $\mathcal{G}$ is the set $\{w \mid \text{sg}(w) \in L(\mathcal{G})\}$.*

One expects that, normally, string languages generated by hypergraph $\mathcal{L}$ grammars should be the same as languages generated by string $\mathcal{L}$ grammars. For example, for $\mathcal{L} = \text{MILL1}$, the following proposition holds.

▶ **Proposition 17.** *If a language L is generated by a string MILL1 grammar, then there is a hypergraph MILL1 grammar $\mathcal{G}$ such that $L^{str}(\mathcal{G}) = L$. Conversely, if $\mathcal{G}$ is a hypergraph MILL1 grammar, then $L^{str}(\mathcal{G}) \setminus \{\varepsilon\}$ is generated by some string MILL1 grammar.*

This proposition is almost trivial; the only minor technicality is that hypergraph MILL1 grammars assign formulae to nodes while string MILL1 grammars are unable to do so. This technicality is the cause why we need to exclude the empty word in the second part of the proposition. See the proof of this proposition in Appendix A.1.
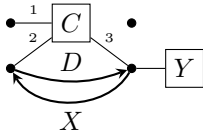
3.1 Hypergraph Transformation Rules as MILL1 Formulae

Our main goal now is to describe a relation between hypergraph first-order linear logic grammars and hypergraph transformation systems. We start with showing how a ht-rule is encoded by a MILL1 formula. Let us fix a unary predicate $\nu(x)$; informally, it is understood as "x is a node". Given a hypergraph H, let us treat its hyperedge labels as predicate symbols. Namely, if $lab_H(e) = a$, then let us assume that the elements of $\text{type}(a)$ are enumerated, i.e. $\text{type}(a) = \{\sigma_1, \ldots, \sigma_n\}$; for any function $h : \text{type}(A) \to \text{Var}$, let $lab_H(e)[h]$ stand for the formula $a(h(\sigma_1), \ldots, h(\sigma_n))$. The arity of a is $n = |\text{type}(a)|$.

▶ **Definition 18.** *The* diagram *of a hypergraph H is the multiset*

$$\mathcal{D}(H) := \{lab_H(e)[att_H(e)] \mid e \in E_H\} \cup \{\nu(v) \mid v \in V_H\}.$$

Let $D(H) := \exists \vec{v} \bigotimes \mathcal{D}(H)$ where $\vec{v}$ is the list of nodes in $V_H \setminus \text{ran}(ext_H)$.

▶ **Example 19.** Let $H = $ be a hypergraph such that $V_H = \{v_1, v_2, v_3, v_4\}$.

Then $\mathcal{D}(H) = C(v_1, v_3, v_4), D(v_3, v_4), X(v_4, v_3), Y(v_4), \nu(v_1), \nu(v_2), \nu(v_3), \nu(v_4)$.

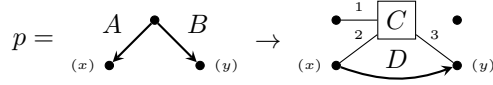▶ **Definition 20.** *Given a ht-rule $p = (H \to H')$, let*
$fm(p) := \forall \vec{u} \, (D(H') \multimap D(H)[\chi_p])$ *where*

- $\vec{u}$ *is the list of nodes in $\text{ran}(ext_{H'})$;*
- χ_p *is a substitution function defined on $\text{ran}(ext_H)$ as follows: $\chi_p(ext_H(\sigma)) = ext_{H'}(\sigma)$.*

The formula $fm(p)$ is closed. Note that χ_p is well defined because ext_H is injective.

▶ **Example 21.** Consider the ht-rule

Let u_1, u_2, u_3 be the nodes of the left-hand side hypergraph (from left to right) and let v_1, v_2, v_3, v_4 be the nodes of the right-hand side hypergraph. Then

$$fm(p) = \forall v_3.\forall v_4.\big((\exists v_1.\exists v_2.C(v_1, v_3, v_4) \otimes D(v_3, v_4) \otimes \nu(v_1) \otimes \nu(v_2) \otimes \nu(v_3) \otimes \nu(v_4)) \multimap$$
$$(\exists u_2.A(u_2, v_3) \otimes B(u_2, v_4) \otimes \nu(v_3) \otimes \nu(u_2) \otimes \nu(v_4))\big).$$

The main lemma about $fm(p)$ is presented below.

▶ **Lemma 22.** *Let P, P' be multisets of ht-rules and let G, G' be two hypergraphs with injective $ext_G, ext_{G'}$. The sequent*

$$\{!fm(r) \mid r \in P\}, \{fm(r) \mid r \in P'\}, \mathcal{D}(G') \vdash D(G)[\chi_{G \to G'}] \tag{1}$$

is derivable in ILL1 *if and only if there exists a derivation of a hypergraph isomorphic to G' from G which uses each rule from P' exactly once and that can use rules from P any number of times.*

The proof of this lemma, placed in Appendix A.2, is quite technical; its idea is to do a proof search using focusing [1]. Using Lemma 22 we can prove the following theorem.

▶ **Theorem 23.** *Hypergraph* ILL1 *grammars generate the class of all recursively enumerable hypergraph languages.*

Proof. Clearly, all languages generated by hypergraph ILL1 grammars are recursively enumerable. To prove the converse, according to Proposition 9, it suffices to show that any ht-system $\mathcal{G} = \langle N, T, P, S \rangle$ can be converted into a hypergraph ILL1 grammar generating the same language. Define the hypergraph ILL1 grammar $\mathcal{G}' = \langle T, S', \text{type}(S), \triangleright \rangle$ such that
1. $S' := \bigotimes\{!fm(r) \mid r \in P\} \multimap D(S)[h_S]$ where $h_S(ext_S(\sigma)) = \sigma$ for $\sigma \in \text{type}(S)$;
2. $a \triangleright a(\sigma_1, \ldots, \sigma_n)$ for $a \in T$ and $\text{type}(a) = \{\sigma_1, \ldots, \sigma_n\}$;
3. $\bullet \triangleright \nu(x_\bullet)$.
A hypergraph H is accepted by $\mathcal{G}'$ if and only if the following sequent is derivable in ILL1:

$$\{lab_H(e)[att_H(e)] \mid e \in E_H\}, \{\nu(v) \mid v \in V_H\} \vdash S'[ext_H].$$

Note that the antecedent of this sequent is the diagram of H and that the succedent equals $\bigotimes\{!fm(r) \mid r \in P\} \multimap D(S)[h_S][ext_H] = \bigotimes\{!fm(r) \mid r \in P\} \multimap D(S)[\chi_{S \to H}]$. By invertibility of the rules $(\multimap R)$ and $(\otimes L)$, the above sequent is equiderivable with the one

$$\{!fm(r) \mid r \in P\}, \mathcal{D}(H) \vdash D(S)[\chi_{S \to H}]$$

By Lemma 22, derivability of the latter sequent is equivalent to the fact that H is derivable from S using rules from P, i.e. that $H \in L(\mathcal{G})$. Thus, $L(\mathcal{G}) = L(\mathcal{G}')$. ◀

This result justifies soundness of Definition 12: if hypergraph grammars based on ILL1, which includes the powerful exponential modality, were not Turing-complete, this would indicate that we do not have enough control in the grammar formalism. For example, if we did not include node-formula assignment in Definition 12, then Theorem 23 would be false.

3.2 Hypergraph MILL1 Grammars

We proceed to investigating expressive power of hypergraph MILL1 grammars. They are clearly less expressive than ht-systems because they generate only languages from NP. Nevertheless, it turns out that they as powerful as ht-systems in which the length of a derivation of a hypergraph is bounded by a linear function w.r.t. the size of the hypergraph.

▶ **Definition 24.** *The size of a hypergraph H denoted by $|H|$ is the number of nodes and hyperedges in H. A* linear-time hypergraph transformation system *is a ht-system $\mathcal{G} = \langle N, T, P, S \rangle$ for which there is $c \in \mathbb{N}$ (a* time constant*) such that, for each $H \in L(\mathcal{G})$, there is a derivation $S \Rightarrow^* H$ with at most $c \cdot |H|$ steps.*

Adding linear-time bound to formal grammars has a long history. Linear-time type-0 Chomsky grammars were studied in [2, 11]; linear-time one-tape Turing machines were studied in [34]; linear-time branching vector addition systems were studied in [29] in the context of commutative Lambek grammars. However, to our best knowledge, linear-time graph grammars have not appeared in the literature. Linear-time ht-systems may be considered as the hypergraph counterpart of linear-time type-0 grammars studied in [2, 11], so it is quite a natural formalism. The main result concerning them is presented below.

▶ **Theorem 25.** *Each linear-time ht-system can be converted into an equivalent hypergraph* MILL1 *grammar.*

Proof of Theorem 25. Let $\mathcal{G} = \langle N, T, P, S \rangle$ be a linear-time ht-system with the time constant $c \in \mathbb{N}$. Define the hypergraph MILL1 grammar $\mathcal{G}' = \langle T, S', \text{type}(S), \triangleright \rangle$ such that
1. $S' := D(S)[h_S]$ where $h_S(ext_S(\sigma)) = \sigma$ for $\sigma \in \text{type}(S)$;
2. $a \triangleright a(\sigma_1, \ldots, \sigma_n) \otimes fm(p_1) \otimes \ldots \otimes fm(p_k)$ for $a \in T$, $\text{type}(a) = \{\sigma_1, \ldots, \sigma_n\}$, $0 \le k \le c$, and for $p_1, \ldots, p_k \in P$;
3. $\bullet \triangleright \nu(x_\bullet) \otimes fm(p_1) \otimes \ldots \otimes fm(p_k)$ for $0 \le k \le c$, and for $p_1, \ldots, p_k \in P$.
A hypergraph H is accepted by $\mathcal{G}'$ if and only if, for each $e \in E_H$ (and for each $v \in V_H$), there exist at most c rules from P, which we denote by $p_1^e, \ldots, p_{k(e)}^e$ (by $q_1^v, \ldots, q_{k(v)}^v$ resp.), such that the sequent

$$\{lab_H(e)[att_H(e)] \otimes fm(p_1^e) \otimes \ldots \otimes fm(p_{k(e)}^e) \mid e \in E_H\},$$
$$\{\nu(v) \otimes fm(q_1^v) \otimes \ldots \otimes fm(q_{k(v)}^v) \mid v \in V_H\} \vdash S'[ext_H]$$

is derivable in MILL1. By invertibility of $(\otimes L)$, this is equivalent to the fact that there exists a multiset P' of cardinality at most $c \cdot |E_H| + c \cdot |V_H| = c \cdot |H|$ such that all its elements are from P and such that the following sequent is derivable in MILL1:

$$\{fm(p) \mid p \in P'\}, \mathcal{D}(H) \vdash D(S)[\chi_{S \to H}]$$

By Lemma 22, this is equivalent to the fact that there is a derivation of a hypergraph H' isomorphic to H from S that uses each rule from P' exactly once. Existence of P' satisfying these properties is equivalent to the fact that there is a derivation of H' from S of length at most $c \cdot |H|$, which is equivalent to $H \in L(\mathcal{G})$. Thus, $L(\mathcal{G}) = L(\mathcal{G}')$. ◀

The question arises whether the converse holds as well. We are not going to address it in the article because of space-time limitations and also because we are mainly interested in lower bounds for the class of hypergraph MILL1 grammars. Still, we claim that it is possible to convert each hypergraph MILL1 grammar into a linear-time ht-system *with non-injective rules*, i.e. with rules $H \to H'$ where $ext_{H'}$ is allowed to be non-injective. This could be done

by a straightforward (yet full of tiring technical details) encoding of MILL1 inference rules by hypergraph transformations. We leave proving that for the future work.

Speaking of upper bounds, we noted that all languages generated by hypergraph MILL1 grammars are in NP; besides, as we shall show in Theorem 30, there is an NP-complete language of string graphs generated by a hypergraph MILL1 grammars, so the NP upper bound is accurate.

Let us further explore properties of the class of languages generated by hypergraph MILL1 grammars. It turns out to be closed under intersection.

▶ **Theorem 26.** *Languages generated by hypergraph* MILL1 *grammars are closed under intersection.*

Proof. Let $\mathcal{G}_i = \langle T_i, S_i, X_i, \rhd_i \rangle$ be two hypergraph MILL1 grammars; we aim to construct a hypergraph MILL1 grammar generating $L(\mathcal{G}_1) \cap L(\mathcal{G}_2)$. Let us assume without loss of generality that $T_1 = T_2 = T$; also, let us assume that predicate symbols used by $\mathcal{G}_1$ and $\mathcal{G}_2$ are disjoint. Let us denote the set of subformulas of formulas occuring in $\mathcal{G}_i$ by Fm_i $(i = 1, 2)$. Note that, if $X_1 \neq X_2$, then $L(\mathcal{G}_1) \cap L(\mathcal{G}_2) = \emptyset$ (hypergraphs in these languages would have different types); thus, we can assume that $X_1 = X_2$ (let us denote this set by X).

Define the grammar $\mathcal{G} := \langle T, S, X, \rhd \rangle$ where $S = S_1 \otimes S_2$ and $\rhd$ is the smallest relation such that $a \rhd A_1 \otimes A_2$ holds whenever $a \in T \cup \{\bullet\}$ and $a \rhd_i A_i$ for $i = 1, 2$.

▶ **Lemma 27** (splitting lemma)**.**
1. *If the sequent $A_1, \ldots, A_n \vdash B$ is derivable in* MILL1 *such that $A_i \in \mathrm{Fm}_1 \cup \mathrm{Fm}_2$ and $B \in \mathrm{Fm}_k$ for some $k \in \{1, 2\}$, then all A_i are also from Fm_k.*
2. *The sequent $A_1, \ldots, A_n, B_1, \ldots, B_m \vdash A \otimes B$ such that A_i and A are from Fm_1 and B_i, B are from Fm_2 is derivable in* MILL1 *if and only if the sequents $A_1, \ldots, A_n \vdash A$ and $B_1, \ldots, B_m \vdash B$ are derivable in* MILL1.

Both statements are proved jointly by straightforward induction on the length of a derivation.

A hypergraph H is accepted by $\mathcal{G}$ if and only if, for $i = 1, 2$, there exist functions $h_V^i : V_H \to \mathrm{Fm}_i$ and $h_E^i : E_H \to \mathrm{Fm}_i$ such that $\bullet \rhd_i h_V^i(v)$ for $v \in V_H$, $lab_H(e) \rhd_i h_E^i(v)$ for $e \in E_H$, and the following sequent is derivable in MILL1:

$$\{(h_E^1(e) \otimes h_E^2(e))[att_H(e)] \mid e \in E_H\}, \{(h_V^1(v) \otimes h_V^2(v))[v/x_\bullet] \mid v \in V_H\} \vdash (S_1 \otimes S_2)[ext_H].$$

By invertibility of $(\otimes L)$ and splitting lemma, this is equivalent to derivability of the sequents $\{h_E^i(e)[att_H(e)] \mid e \in E_H\}, \{h_V^i(v)[v/x_\bullet] \mid v \in V_H\} \vdash S_i[ext_H]$ for $i = 1, 2$, hence is equivalent to the fact that H belongs to $L(\mathcal{G}_1) \cap L(\mathcal{G}_2)$.   ◀

An analogous technique cannot be used for Lambek categorial grammars because the Lambek calculus is non-commutative, and splitting lemma does not hold for it. In fact, since Lambek categorial grammars generate context-free languages [25], which are not closed under intersection, a similar result does not hold for Lambek categorial grammars.

4   Expressive Power of String MILL1 Grammars

Now, let us apply the results and techniques developed for hypergraph MILL1 grammars to describing the class of languages generated by string MILL1 grammars. First, Proposition 17 and Theorem 25 imply the following lower bound.

▶ **Corollary 28.** *If $\mathcal{G} = \langle N, T, P, S \rangle$ is a linear-time ht-system, then $\{w \in T^+ \mid \mathrm{sg}(w) \in L(\mathcal{G})\}$ is generated by a string* MILL1 *grammar.*

Next, the following result can be proved in the same way as Theorem 26.

▶ **Theorem 29.** *Languages generated by string MILL1 grammars are closed under intersection.*

(Note that we cannot directly infer this theorem from Proposition 17 and Theorem 26 because of the empty string issue.) Since languages generated by string MILL1 grammars contain all context-free languages, they also contain their finite intersections, in particular, the language

$$\{(a^n b^n)^n \mid n > 0\} = \{a^{n_1} b^{n_1} \ldots a^{n_k} b^{n_k} \mid n_1, \ldots, n_k \in \mathbb{N}\}$$
$$\cap \{a^k b^{n_1} a^{n_1} \ldots b^{n_{k-1}} a^{n_{k-1}} b^l \mid k, l, n_1, \ldots, n_{k-1} > 0\}.$$

It is simple to prove that languages generated by string MILL1 grammars are also closed under letter-to-letter homomorphisms, so the language $\{a^{2n^2} \mid n \in \mathbb{N}\}$ can be generated by a string MILL1 grammar as well. Thus, string MILL1 grammars generate languages with non-semilinear Parikh images.

Note that, for Lambek categorial grammars, there is imbalance between expressive power and algorithmic complexity. Namely, on the one hand, Lambek categorial grammars generate exactly context-free languages, all of which are polynomially parsable, but on the other hand, parsing in the Lambek calculus is an NP-complete problem [27]. This is not the case for MILL1 grammars, as the following theorem shows.

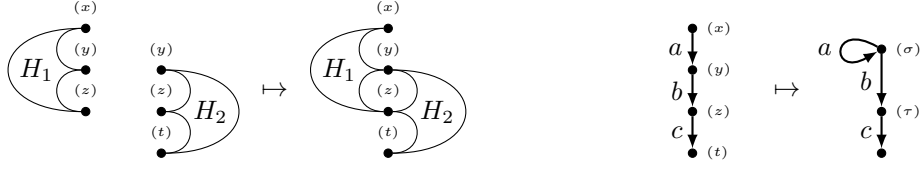▶ **Theorem 30.** *String* MILL1 *grammars generate an NP-complete language.*

A straightforward proof of this theorem can be found in Appendix A.3. There, we present a linear-time ht-system generating an NP-complete language of string graphs. Here, we shall present another proof that relies on a result from [3].

Proof. Consider a string rewriting system $\mathcal{S} = \langle N, T, P, S \rangle$ where P is a finite set of rules of the form $\alpha \to \beta$ where $\alpha, \beta \in (N \cup T)^*$ are arbitrary strings and $S \in (N \cup T)^*$ is the start string. Recall that $\eta \alpha \theta \Rightarrow_{\mathcal{S}} \eta \beta \theta$ whenever $(\alpha \to \beta) \in P$ (and no other pairs of strings are in the relation $\Rightarrow_{\mathcal{S}}$); recall also that $L(\mathcal{S}) = \{w \in T^* \mid S \Rightarrow_{\mathcal{S}}^* w\}$.

Let us convert $\mathcal{S}$ into the ht-system $\mathcal{G} = \langle N, T, P', \mathrm{sg}(S) \rangle$ where the symbols from $N \cup T$ have the type $\{\mathbf{s}, \mathbf{t}\}$ and $P' = \{\mathrm{sg}(\alpha) \to \mathrm{sg}(\beta) \mid (\alpha \to \beta) \in P\}$. (Let us assume that, in each rule $\alpha \to \beta$, α and β are nonempty.) It is straightforward to check that $L(\mathcal{G}) = \{\mathrm{sg}(w) \mid w \in L(\mathcal{S})\}$. Finally, [3, Theorem 1] says that there is a linear-time string rewriting system $\mathcal{S}$ that generates an NP-complete language. Thus, if one constructs $\mathcal{G}$ from it and then applies Corollary 28, then they obtain a string MILL1 grammar generating an NP-complete language.   ◀

One of the reviewers pointed out that string MILL1 grammars generating nonsemilinear languages are too much for linguistic applications, where it is widely assumed that natural languages are semilinear. The reviewer asked whether restricting MILL1 to the fragment where each (free or bound) variable occurs in each formula at most twice results in languages generated by the corresponding class of grammars being semilinear. I am afraid that, even with this restriction, string MILL1 grammars generate some non-semilinear and NP-complete languages. Let us consider the above construction from Theorem 30. E.g. if $AB \to BCD$ is a string rewriting rule, then

$$fm(\mathrm{sg}(AB) \to \mathrm{sg}(BCD)) = \forall v_1. \forall v_2. \big[(\exists x. A(v_1, x) \otimes B(x, v_2) \otimes \nu(v_1) \otimes \nu(x) \otimes \nu(v_2)) \multimap$$
$$(\exists y. \exists z. B(v_1, y) \otimes C(y, z) \otimes D(z, v_2) \otimes \nu(v_1) \otimes \nu(y) \otimes \nu(z) \otimes \nu(v_2)) \big].$$

Figure 3 Scheme of parallel composition of H_1 and H_2 (left) and an example of substitution of a hypergraph according to h where $h(x) = h(y) = \sigma$, $h(z) = \tau$, and $h(t)$ is undefined (right).

In a formula of the form $fm(\mathrm{sg}(\alpha) \to \mathrm{sg}(\beta))$, each variable occurs at most four times. However, we claim that one could remove ν predicates everywhere, which would result in each variable occurring in each formula at most twice. We also claim that Lemma 22 would remain true without ν predicates in formulas in case where we consider only string graphs. Thus, we can use Book's result from [3] to generate an NP-complete and a non-semilinear language by a grammar over the restricted fragment of MILL1. We do not provide formal proofs but leave them for the future work.

5   Hypergraph Language Semantics

Now, let us proceed to model-theoretic investigations into MILL1. Our objective is to generalise language models for the Lambek calculus to MILL1 and to devise *hypergraph language models*. This will enable one to regard MILL1 as a logic for reasoning about hypergraph resources. The most important question is how the composition of such resources should be defined: if H_1, H_2 are two hypergraphs, then how should one understand "$H_1 \otimes H_2$"?

Language semantics for the Lambek calculus, algebraically speaking, is a mapping of L formulae to a free semigroup of words which interprets product $A \cdot B$ as elementwise product of interpretations of A and B. What is the hypergraph counterpart of free semigroups? In the field of hyperedge replacement, there are algebras of hypergraphs called HR-algebras [6, 7, 30]. They include the parallel composition operation and source manipulating operations. Parallel composition is a way of gluing hypergraphs defined as follows.

▶ **Definition 31.** *Let H_1 and H_2 be hypergraphs. Let $\sim$ be the smallest equivalence relation on $V_{H_1} \sqcup V_{H_2}$ such that $ext_{H_1}(\sigma) \sim ext_{H_2}(\sigma)$ for $\sigma \in \mathrm{type}(H_1) \cap \mathrm{type}(H_2)$. Then,* parallel composition $H_1 \parallel H_2$ *is the hypergraph H such that* $\mathrm{type}(H) = \mathrm{type}(H_1) \cup \mathrm{type}(H_2)$; $V_H = (V_{H_1} \sqcup V_{H_2})/\sim$, $E_H = E_{H_1} \sqcup E_{H_2}$; $att_H(e)(\sigma) = [att_{H_i}(e)(\sigma)]_\sim$, $lab_H(e) = lab_{H_i}(e)$ *for* $e \in E_{H_i}$; $ext_H(\sigma) = [ext_{H_i}(\sigma)]_\sim$ *for* $\sigma \in \mathrm{type}(H_1) \cup \mathrm{type}(H_2)$.

Informally, $H_1 \parallel H_2$ is obtained by taking the disjoint union of H_1 and H_2 and fusing $ext_{H_1}(\sigma)$ with $ext_{H_2}(\sigma)$ for $\sigma \in \mathrm{type}(H_1) \cap \mathrm{type}(H_2)$. This operation is illustrated on Figure 3. Note that parallel composition is associative and commutative.

Other operations used in HR-algebras allow one to reassign selectors to external nodes, to make an external node non-external, or to fuse some external nodes. We shall use an operation that unifies all the three manipulations; we shall call it *substitution*.

▶ **Definition 32.** *Given a hypergraph H and a partial function $h \subseteq \Sigma \times \Sigma$, let $\sim$ be the smallest equivalence relation such that $ext_H(\sigma_1) \sim ext_H(\sigma_2)$ whenever $h(\sigma_1) = h(\sigma_2)$. Then $sub_h(H)$ is the hypergraph H' such that* $\mathrm{type}(H') = h(\mathrm{type}(H))$; $V_{H'} = V_H/\sim$; $E_{H'} = E_H$; $att_{H'}(e)(\sigma) = [att_H(e)(\sigma)]_\sim$, $lab_{H'}(e) = lab_H(e)$ *for* $e \in E$; $ext_{H'}(h(\sigma)) = [ext_H(\sigma)]_\sim$.

One can compare the substitution operation with the operations of source renaming and source fusion from [30] and verify that the former one is interdefinable with the latter ones

(in presence of parallel composition). Hence, one can use $/\!\!/$ and sub_h as basic operations of HR-algebras. Nicely, exactly these operations can also be used for defining hypergraph language models, which we are going to do now. Let $K_0 := \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$ be the empty hypergraph. From now on, $\Sigma = \mathrm{Var}$ (i.e. selectors and variables are the same objects).

▶ **Definition 33.** *A hypergraph language model is a pair $\langle T, u \rangle$ where T is a Σ-typed alphabet and $u : \mathrm{Fm}(\mathrm{MILL1}) \to \mathcal{P}(\mathcal{H}(T))$ is a function mapping formulas of* $\mathrm{MILL1}$ *to sets of abstract hypergraphs over T which satisfies the following conditions:*

1. *$sub_h(u(A)) \subseteq u(A[h])$ for any total function $h : \Sigma \to \Sigma$;*
2. *$u(A \otimes B) = u(A) /\!\!/ u(B)$;*
3. *$u(A \multimap B) = \{H \mid \forall H' \in u(A) \ (H /\!\!/ H' \in u(B))\}$;*
4. *$u(\exists x A) = \bigcup\limits_{y \in \mathrm{Var}} u(A[y/x])$;*
5. *$u(\forall x A) = \bigcap\limits_{y \in \mathrm{Var}} u(A[y/x])$.*

A sequent $A_1, \ldots, A_n \vdash B$ is true in this model if $u(A_1) /\!\!/ \ldots /\!\!/ u(A_n) \subseteq u(B)$ (for $n = 0$, if $K_0 \in u(B)$).

This semantics can be viewed as an instance of intuitionistic phase semantics [15] with the commutative monoid $(\mathcal{H}(T), /\!\!/, K_0)$ and with the trivial closure operator $\mathrm{Cl}(X) = X$.

The first property of u in Definition 33 relates substitution as a logical operation to substitution as a hypergraph transformation; without it, $u(A)$ and $u(A[h])$ would be unrelated, which is undesirable. Besides, this property is used to prove correctness. Note that, if $h : \Sigma \to \Sigma$ is a bijection, then $u(A[h]) = sub_h(u(A))$ (apply property 1 twice). Quantifiers are interpreted as additive conjunction and disjunction, which reflects their behaviour correctly.

▶ **Lemma 34.**
1. *$u(A) /\!\!/ u(B) \subseteq u(C)$ if and only if $u(A) \subseteq u(B \multimap C)$.*
2. *If $u(A) \subseteq u(B)$, then $u(A[h]) \subseteq u(B[h])$ for any substitution h.*

Proof. 1. Trivially follows from the definition of a model.

2. Clearly, $K_0 /\!\!/ H = H$ for any H. Therefore, $u(A) \subseteq u(B)$ implies $K_0 \in u(A \multimap B)$. Consequently, according to item 1 of Definition 33, $K_0 = sub_h(K_0) \in u(A[h] \multimap B[h])$. This implies that $u(A[h]) \subseteq u(B[h])$, as desired. ◀

The main result is soundness of MILL1 and completeness of its fragment $\mathrm{MILL1}(\multimap, \forall)$ w.r.t. hypergraph language models.

▶ **Theorem 35.**
1. $\mathrm{MILL1}$ *is sound w.r.t. hypergraph language models.*
2. $\mathrm{MILL1}(\multimap, \forall)$ *is complete w.r.t. hypergraph language models.*

Proof. Soundness can be checked straightforwardly by showing that the conclusion of each rule with true premises is also true. The only nontrivial cases are the rules $(\exists L)$ and $(\forall R)$. Assume that $\Gamma, A[z/x] \vdash B$ where z does not occur freely in Γ, B is true in a model $\langle T, u \rangle$. Without loss of generality, let us assume that Γ is empty (otherwise, we can move it to the succeedent using Lemma 34). Under this assumption, we are given that $u(A[z/x]) \subseteq u(B)$. By Lemma 34, for each $y \in \mathrm{Var}$, $u(A[y/x]) = u(A[z/x][y/z]) \subseteq u(B[y/z]) = u(B)$. This proves that $u(\exists x A) \subseteq u(B)$. The case $(\forall R)$ is dealt with similarly.

Completeness of $\mathrm{MILL1}(\multimap, \forall)$ is proved by constructing a canonical model $\langle \mathrm{T}, \mathrm{u} \rangle$. Before we do this, let us introduce a new notion used in the construction.

▶ **Definition 36.** *Assume that a countable sequence of variables $\xi_1, \xi_2, \ldots$ is fixed. Given a formula A, let us read it from left to right and replace the i-th occurrence of a free variable from the left by ξ_i. We denote the resulting formula by A°.*

For example, $(\exists x(A(x,y,y) \otimes \forall z B(z,y,t)))^\circ = \exists x(A(x,\xi_1,\xi_2) \otimes \forall z B(z,\xi_3,\xi_4))$. We shall use formulas of the form A° as labels of hyperedges in the canonical model with $\mathrm{type}(A^\circ) = \mathrm{FVar}(A^\circ)$. So, $\mathrm{T} := \{A^\circ \mid A \in \mathrm{Fm}(\mathrm{MILL1}(\multimap, \forall))\}$. The idea is that, in the canonical model, variables are represented by nodes, while hyperedge labels do not carry information about variables ($\xi_1, \xi_2, \ldots$ are, informally, placeholders for variables).

Given a sequent $\Gamma \vdash A$ where $\Gamma = A_1, \ldots, A_n$ ($n \geq 0$) and given two finite sets of variables X, Y such that $Y \subseteq \mathrm{FVar}(\Gamma) \cup X$, let us define the hypergraph $H_{\Gamma;A}^{X;Y} = \langle V, E, att, lab, ext \rangle$ as follows: $V = \mathrm{FVar}(\Gamma) \cup X$; $E = \{e_1, \ldots, e_n\}$; $att(e_i)(x) = h_i(x)$ for $x \in \mathrm{FVar}(A_i^\circ)$ where $h_i : \Sigma \to \Sigma$ is a function such that $A_i = A_i^\circ[h_i]$; $lab(e_i) = A_i^\circ$; $ext(x) = x$ for $x \in (V \cap \mathrm{FVar}(A)) \cup Y$ (and undefined otherwise).

Let $\mathrm{u}(A) := \{H_{\Gamma;A}^{X;Y} \mid \Gamma \vdash A \text{ is derivable in MILL1}(\multimap, \forall)\}$. The proof that $\langle \mathrm{T}, \mathrm{u} \rangle$ is a hypergraph language model is quite technical and it can be found in Appendix A.4. Now, assume that $A_1, \ldots, A_n \vdash B$ is true in this model. Then, $K_0 \in \mathrm{u}(A_1 \multimap (\ldots \multimap (A_n \multimap B)))$, i.e. $K_0 = H_{\emptyset; A_1 \multimap (\ldots \multimap (A_n \multimap B))}^{X;Y}$, and $\vdash A_1 \multimap (\ldots \multimap (A_n \multimap B))$ is derivable in MILL1$(\multimap, \forall)$. By invertibility of $(\multimap R)$, the sequent $A_1, \ldots, A_n \vdash B$ is derivable. ◀

6   Conclusion

As we have shown, first-order intuitionistic linear logic does have strong connections to the hypergraph grammar theory, namely, to hypergraph transformation systems and to HR-algrebras. The notion of hypergraph first-order categorial grammars naturally and simply extends the concept of Lambek categorial grammars to hypergraphs. Developing hypergraph MILL1 grammars and relating them to hypergraph transformation systems gave us useful insights into expressive power of string MILL1 grammars. In turn, the notion of a hypergraph language model revealed a previously unknown connection of first-order intuitionistic linear logic to the apparatus of HR-algebras, the latter having been studied mainly in the context of monadic second-order definability.

Several questions remain open for the future work. The first one is whether the converse to Theorem 25 holds; more generally, it is desirable to characterise precisely hypergraph MILL1 grammars in terms of hypergraph transformation systems. The second one is whether string MILL1 grammars over the restricted fragment where each variable is allowed to be used in a formula at most twice generate only nonsemilinear languages (I explained earlier why I think that the answer to this question is negative). The third one is whether MILL1 is complete w.r.t. hypergraph language models.

───── **References** ─────

1   Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992. `doi:10.1093/logcom/2.3.297`.

2   Ronald V. Book. Time-bounded grammars and their languages. *Journal of Computer and System Sciences*, 5(4):397–429, 1971. `doi:10.1016/S0022-0000(71)80025-9`.

3   Ronald V. Book. On the complexity of formal grammars. *Acta Informatica*, 9:171–181, 1978. `doi:10.1007/BF00289076`.

4   Wojciech Buszkowski. Compatibility of a categorial grammar with an associated category system. *Mathematical Logic Quarterly*, 28(14-18):229–238, 1982. `doi:10.1002/malq.19820281407`.

5   Wojciech Buszkowski. *Type Logics in Grammar*, pages 337–382. Springer Netherlands, Dordrecht, 2003. `doi:10.1007/978-94-017-3598-8_12`.

6   Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990. `doi:10.1016/0890-5401(90)90043-H`.

7   Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2012.

8   Frank Drewes, Hans-Jörg Kreowski, and Annegret Habel. Hyperedge replacement graph grammars. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*, pages 95–162. World Scientific, 1997. `doi:10.1142/9789812384720\_0002`.

9   Joost Engelfriet. Context-free graph grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Volume 3: Beyond Words*, pages 125–213. Springer, 1997. `doi:10.1007/978-3-642-59126-6\_3`.

10   Jean-Yves Girard. Linear logic: A survey. In Friedrich L. Bauer, Wilfried Brauer, and Helmut Schwichtenberg, editors, *Logic and Algebra of Specification*, pages 63–112, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg. `doi:10.1007/978-3-642-58041-3_3`.

11   Aleksei Gladkii. On complexity of inference in phase-structure grammars. *Algebra i Logika. Sem. (in Russian)*, 3(5-6):29–44, 1964.

12   Annegret Habel. *Hyperedge Replacement: Grammars and Languages*, volume 643 of *Lecture Notes in Computer Science*. Springer, 1992. `doi:10.1007/BFB0013875`.

13   Annegret Habel, Jürgen Müller, and Detlef Plump. Double-pushout graph transformation revisited. *Mathematical Structures in Computer Science*, 11(5):637–688, 2001. `doi:10.1017/S0960129501003425`.

14   Laura Kallmeyer. *Parsing Beyond Context-Free Grammars.* Springer Berlin Heidelberg, 2010. `doi:10.1007/978-3-642-14846-0`.

15   Max I. Kanovich, Mitsuhiro Okada, and Kazushige Terui. Intuitionistic phase semantics is almost classical. *Mathematical. Structures in Comp. Sci.*, 16(1):67–86, 2006. `doi:10.1017/S0960129505005062`.

16   Yuichi Komori. Predicate logics without the structure rules. *Studia Logica*, 45(4):393–404, 1986. `doi:10.1007/bf00370272`.

17   Barbara König, Dennis Nolte, Julia Padberg, and Arend Rensink. A tutorial on graph transformation. In Reiko Heckel and Gabriele Taentzer, editors, *Graph Transformation, Specifications, and Nets - In Memory of Hartmut Ehrig*, volume 10800 of *Lecture Notes in Computer Science*, pages 83–104. Springer, 2018. `doi:10.1007/978-3-319-75396-6\_5`.

18   Joachim Lambek. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):154–170, 1958. `doi:10.1080/00029890.1958.11989160`.

19   Michael Moortgat. Multimodal linguistic inference. *Journal of Logic, Language and Information*, 5(3–4):349–385, 1996. `doi:10.1007/bf00159344`.

20   Richard Moot. Extended Lambek calculi and first-order linear logic. In Claudia Casadio, Bob Coecke, Michael Moortgat, and Philip J. Scott, editors, *Categories and Types in Logic, Language, and Physics - Essays Dedicated to Jim Lambek on the Occasion of His 90th Birthday*, volume 8222 of *Lecture Notes in Computer Science*, pages 297–330. Springer, 2014. `doi:10.1007/978-3-642-54789-8\_17`.

21   Richard Moot. Hybrid type-logical grammars, first-order linear logic and the descriptive inadequacy of lambda grammars, 2014. URL: `https://arxiv.org/abs/1405.6678`, `arXiv:1405.6678`.

22   Richard Moot and Mario Piazza. Linguistic applications of first order intuitionistic linear logic. *Journal of Logic, Language and Information*, 10(2):211–232, 2001. `doi:10.1023/a:1008399708659`.

23   Richard Moot and Christian Retoré. *The Logic of Categorial Grammars - A Deductive Account of Natural Language Syntax and Semantics*, volume 6850 of *Lecture Notes in Computer Science*. Springer, 2012. `doi:10.1007/978-3-642-31555-8`.

24   Glyn Morrill, Oriol Valentín, and Mario Fadda. The displacement calculus. *Journal of Logic, Language and Information*, 20(1):1–48, 2010. `doi:10.1007/s10849-010-9129-2`.

25   Mati Pentus. Lambek grammars are context free. In *Proceedings Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 429–433, 1993. `doi:10.1109/LICS.1993.287565`.

26   Mati Pentus. Models for the Lambek calculus. *Ann. Pure Appl. Log.*, 75(1-2):179–213, 1995. `doi:10.1016/0168-0072(94)00063-9`.

27   Mati Pentus. Lambek calculus is NP-complete. *Theoretical Computer Science*, 357(1):186–201, 2006. Clifford Lectures and the Mathematical Foundations of Programming Semantics. `doi:10.1016/j.tcs.2006.03.018`.

28   Tikhon Pshenitsyn. Hypergraph Lambek grammars. *Journal of Logical and Algebraic Methods in Programming*, 129:100798, 2022. `doi:10.1016/j.jlamp.2022.100798`.

29   Tikhon Pshenitsyn. Commutative Lambek grammars. *Journal of Logic, Language and Information*, 32(5):887–936, 2023. `doi:10.1007/s10849-023-09407-z`.

30   Grzegorz Rozenberg. *Handbook of Graph Grammars and Computing by Graph Transformation*. World Scientific, 1997. `doi:10.1142/3303`.

31   Sylvain Salvati. A note on the complexity of abstract categorial grammars. In Christian Ebert, Gerhard Jäger, and Jens Michaelis, editors, *The Mathematics of Language*, pages 266–271, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

32   Harold Shellinx. Some syntactical observations on linear logic. *Journal of Logic and Computation*, 1(4):537–559, 09 1991. `doi:10.1093/logcom/1.4.537`.

33   Sergey Slavnov. Making first order linear logic a generating grammar. *Logical Methods in Computer Science*, Volume 19, Issue 4, 2023. `doi:10.46298/lmcs-19(4:11)2023`.

34   Kohtaro Tadaki, Tomoyuki Yamakami, and Jack C.H. Lin. Theory of one-tape linear-time Turing machines. *Theoretical Computer Science*, 411(1):22–43, 2010. `doi:10.1016/j.tcs.2009.08.031`.

35   Tadahiro Uesu. A system of graph grammars which generates all recursively enumerable sets of labelled graphs. *Tsukuba Journal of Mathematics*, 2(none):11 − 26, 1978. `doi:10.21099/tkbjm/1496158502`.

36   Johan van Benthem. *Language in Action: Categories, Lambdas and Dynamic Logic*. MIT Press, 1995.

A Proofs

A.1 Proof of Proposition 17

Let $\mathcal{G} = \langle T, S, \rhd \rangle$ be a string MILL1 grammar. To make a hypergraph MILL1 grammar from it, we need to assign some formula to the node symbol. If there was the unit in our calculus, we could just assign it to nodes, but we do not have it. Let us fix a predicate symbol q of arity 0 not occuring in formulae of $\mathcal{G}$. We define the hypergraph MILL1 grammar $\mathcal{G}' = \langle T, S', \{\mathbf{s}, \mathbf{t}\}, \rhd' \rangle$ as follows:

- $S' := S \otimes (q \multimap q)$;
- for $a \in T$, $a \rhd' A$ iff $a \rhd A$;
- $\bullet \rhd (q \multimap q)$.

A string graph $\mathrm{sg}(a_1 \dots a_n)$ is accepted by $\mathcal{G}'$ if and only if there is a function $h_E : \{e_1, \dots, e_n\} \to \mathrm{Fm}(\mathrm{MILL1})$ such that $a_i \rhd h_E(e_i)$ and the sequent

$$\{h_E(e_i)[x_{i-1}/\mathbf{s}, x_i/\mathbf{t}] \mid i = 1, \dots, n\}, (q \multimap q)^{n+1} \vdash S \otimes (q \multimap q)$$

is derivable in MILL1. By Lemma 27, this is equivalent to derivability of $\{h_E(e_i)[x_{i-1}/\mathbf{s}, x_i/\mathbf{t}] \mid i = 1, \dots, n\} \vdash S$ and to that of $(q \multimap q)^{n+1} \vdash q \multimap q$. The latter sequent is derivable for each n, so this is equivalent to the fact that $a_1 \dots a_n$ belongs to $L(\mathcal{G})$.

The other way around, let $\mathcal{G} = \langle T, S, \{\mathbf{s}, \mathbf{t}\}, \rhd \rangle$ be a hypergraph MILL1 grammar. We fix a unary predicate $\mu(x)$ not occurring in $\mathcal{G}$ and define the string MILL1 grammar $\mathcal{G}' = \langle T, S', \rhd' \rangle$ as follows.

- $S' = \mu(\mathbf{s}) \otimes S$;
- $a \rhd' A$ iff $\mathrm{type}(a) = \{\mathbf{s}, \mathbf{t}\}$ and either $A = B \otimes C[\mathbf{t}/x_\bullet]$ or $A = B \otimes \mu(\mathbf{s}) \otimes C_1[\mathbf{s}/x_\bullet] \otimes C_2[\mathbf{t}/x_\bullet]$ where $a \rhd B$ and $\bullet \rhd C$, $\bullet \rhd C_1$, $\bullet \rhd C_2$. We call a formula of the form $B \otimes \mu(\mathbf{s}) \otimes C_1[\mathbf{s}/x_\bullet] \otimes C_2[\mathbf{t}/x_\bullet]$ *border* and a formula of the form $B \otimes C[\mathbf{t}/x_\bullet]$ *non-border*.

The grammar $\mathcal{G}'$ does not accept the empty string because the sequent $\vdash \mu(x_0) \otimes S[x_0/\mathbf{s}, x_0/\mathbf{t}]$ is not derivable in MILL1. Now, consider a string $w = a_1 \dots a_n$ for $n > 0$. It belongs to $L(\mathcal{G}')$ if and only if there are $A_1, \dots, A_n$ such that $a_i \rhd' A_i$ and the sequent $A_1[x_0/\mathbf{s}, x_1/\mathbf{t}], \dots, A_n[x_{n-1}/\mathbf{s}, x_n/\mathbf{t}] \vdash \mu(x_0) \otimes S[x_0/\mathbf{s}, x_n/\mathbf{t}]$ is derivable in MILL1. If the latter is the case, then, clearly, A_1 is a border formula and, for $i > 1$, A_i is a non-border formula. Now, it is not hard to see that this is equivalent to the fact that $\mathrm{sg}(w)$ belongs to $L(\mathcal{G})$.

A.2 Proof of Lemma 22

To analyse derivability of the sequent of interest we shall use Andreoli's triadic calculus Σ_3 [1]. Note that the logic ILL1 we work with is intuitionistic, while Σ_3 is developed for classical linear logic. However, a well-known result from [32] states that the fragments of (first-order) classical linear logic in the language of ILL1 that do not include the constant $\mathbf{0}$ are conservative over ILL1. Thus we can translate intuitionistic formulas into classical ones (in particular, we translate the implication $A \multimap B$ into $A^\perp \parr B$) and analyse the latter.

We use the exact same notation as in [1] with the only difference that we switch positive and negative atoms. We do not introduce focusing here in detail but refer the reader to [1]. Let us, nevertheless, present the axiom and the inference rules which shall occur in the proof.

$$\frac{}{\Theta : X^\perp \Downarrow X} \ (I_1)$$

$$\frac{\Theta : \Gamma \Uparrow L, A[z/x]}{\Theta : \Gamma \Uparrow L, \forall x A} \ (\forall) \qquad \frac{\Theta : \Gamma \Downarrow A[y/x]}{\Theta : \Gamma \Downarrow \exists x A} \ (\exists)$$

$$\frac{\Theta : \Gamma \Uparrow L, A, B}{\Theta : \Gamma \Uparrow L, A \mathbin{⅋} B} \; (⅋) \qquad \frac{\Theta : \Gamma \Downarrow A \quad \Theta : \Delta \Downarrow B}{\Theta : \Gamma, \Delta \Downarrow A \otimes B} \; (\otimes)$$

$$\frac{\Theta : \Gamma, C \Uparrow L}{\Theta : \Gamma \Uparrow L, C} \; (R \Uparrow) \qquad \frac{\Theta : \Gamma \Uparrow D}{\Theta : \Gamma \Downarrow D} \; (R \Downarrow) \qquad \frac{\Theta : \Gamma \Downarrow E}{\Theta : \Gamma, E \Uparrow} \; (D_1) \qquad \frac{\Theta, E : \Gamma \Downarrow E}{\Theta, E : \Gamma \Uparrow} \; (D_2)$$

In the above rules, X is a positive atom; C's outermost connective is not $⅋, \forall$; D's outermost connective is not $\otimes, \exists$ and D is not a positive atom; E is not a negative atom. (Recall that positive atoms are formulae of the form $p(x_1, \ldots, x_n)$ and negative atoms are formulae of the form $p^\perp(x_1, \ldots, x_n)$.)

▶ **Lemma 37.** *Let*

- *P, P' be multisets of ht-rules;*
- *Θ be a multiset of LL1 formulas and Ξ be a multiset consisting of negative atoms such that, for each $x \in$ Var, $\nu^\perp(x)$ occurs at most once in Θ and at most once in Ξ;*
- *$h : ext_G \to$ Var be a function.*

1. *The sequent*

$$\vdash \{fm^\perp(r) \mid r \in P\} : \{fm^\perp(r) \mid r \in P'\}, \Xi, D(G)[h] \Uparrow \qquad (2)$$

   is derivable in Σ_3 if and only if h is injective, $\Xi = \mathcal{D}^\perp(G')$ for some G' such that $\text{type}(G) = \text{type}(G')$, $h(ext_G(\sigma)) = ext_{G'}(\sigma)$ for $\sigma \in \text{type}(G)$ and such that there is a derivation of a hypergraph isomorphic to G' from G which uses each rule from P' exactly once and that can use rules from P any number of times.

2. *The sequent*

$$\vdash \{fm^\perp(r) \mid r \in P\} : \Theta \Downarrow D(G)[h] \qquad (3)$$

   is derivable in Σ_3 if and only if h is injective and $\Theta = \mathcal{D}^\perp(G')$ for some G' isomorphic to G such that $h(ext_G(\sigma)) = ext_{G'}(\sigma)$ for $\sigma \in \text{type}(G)$.

Proof. Let us denote $\{fm^\perp(r) \mid r \in P\}$ by Φ and $\{fm^\perp(r) \mid r \in P'\}$ by Φ'.

Both statements of the lemma are proved jointly by induction on the length of a derivation. Let us prove their "only if" parts. Essentially, we need to do a proof search. Let us start with considering possible cases of the last rule applied in a derivation of (2).

Case 1.

$$\frac{\vdash \Phi : \Phi', \Xi \Downarrow D(G)[h]}{\vdash \Phi : \Phi', \Xi, D(G)[h] \Uparrow} \; (D_1)$$

The premise is a sequent of the form (3). The induction hypothesis completes the proof.

Case 2.

$$\frac{\vdash \Phi : \{fm^\perp(r) \mid r \in P''\}, \Xi, D(G)[h] \Downarrow fm^\perp(p)}{\vdash \Phi : \{fm^\perp(r) \mid r \in P'\}, \Xi, D(G)[h] \Uparrow} \; (D_1)$$

Here $p = (H_1 \to H_2) \in P'$ and $P'' = P' \setminus \{p\}$. Let us denote $\{fm^\perp(r) \mid r \in P''\}$ by Φ''. It holds that $fm^\perp(p) = \exists \vec{u}(D(H_2) \otimes D^\perp(H_1)[\chi_p])$ where $\vec{u}$ is the list of nodes in $\text{ran}(ext_{H_2})$; therefore, the last steps in the derivation of the above premise must be applications of ($\exists$):

$$\frac{\vdash \Phi : \Phi'', \Xi, D(G)[h] \Downarrow (D(H_2) \otimes D^\perp(H_1)[\chi_p])[f]}{\vdash \Phi : \Phi'', \Xi, D(G)[h] \Downarrow fm^\perp(p)} \; (\exists)^*$$

(Here $(\exists)^*$ means that $(\exists)$ is applied several times.) The function f maps $\text{ran}(ext_{H_2})$ to Var. Now, there are two options how the premise of the above rule application can be obtained.

Case 2a.

$$\frac{\vdash \Phi : \Phi_1'', \Xi_1, D(G)[h] \Downarrow D(H_2)[f] \quad \vdash \Phi : \Phi_2'', \Xi_2 \Downarrow D^\perp(H_1)[\chi_p])[f]}{\vdash \Phi : \Phi'', \Xi, D(G)[h] \Downarrow (D(H_2) \otimes D^\perp(H_1)[\chi_p])[f]} \ (\otimes)$$

Here $\Phi_1'', \Phi_2'' = \Phi''$ and $\Xi_1, \Xi_2 = \Xi$. By the induction hypothesis applied to the leftmost premise (which is of the form (3)), it must be the case that $\Xi_1, D(G)[h]$ consists of negative atoms. However, $D(G)[h]$ is not an atom, so this case is impossible.

Case 2b.

$$\frac{\vdash \Phi : \Phi_1'', \Xi_1 \Downarrow D(H_2)[f] \quad \vdash \Phi : \Phi_2'', \Xi_2, D(G)[h] \Downarrow D^\perp(H_1)[\chi_p][f]}{\vdash \Phi : \Phi'', \Xi, D(G)[h] \Downarrow (D(H_2) \otimes D^\perp(H_1)[\chi_p])[f]} \ (\otimes)$$

By the induction hypothesis, f is injective, $\Phi_1'' = \emptyset$ and $\Xi_1 = \mathcal{D}^\perp(H_2')$ where H_2' is isomorphic to H_2 and $f(ext_{H_2}(\sigma)) = ext_{H_2'}(\sigma)$ for $\sigma \in \text{type}(H_2)$. Consequently, $\Phi_2'' = \Phi''$.

Now, let us analyse possible derivations of the rightmost premise. Let $f' := f \circ \chi_p$. The formula $D^\perp(H_1)[f']$ equals $\forall \vec{v} \, \mathfrak{N} \, \mathcal{D}^\perp(H_1)[f']$ where $\vec{v}$ is the list of nodes in $V_{H_1} \setminus \text{ran}(ext_{H_1})$. Any derivation of the rightmost premise must end in the following way:

$$\frac{\dfrac{\dfrac{\dfrac{\vdash \Phi : \Phi'', \Xi_2, \mathcal{D}^\perp(H_1)[f'][g], D(G)[h] \Uparrow}{\vdash \Phi : \Phi'', \Xi_2, D(G)[h] \Uparrow \mathcal{D}^\perp(H_1)[f'][g]} \ (R \Uparrow)^*}{\vdash \Phi : \Phi'', \Xi_2, D(G)[h] \Uparrow \mathfrak{N} \, \mathcal{D}^\perp(H_1)[f'][g]} \ (\mathfrak{N})^*}{\vdash \Phi : \Phi'', \Xi_2, D(G)[h] \Uparrow D^\perp(H_1)[f']} \ (\forall)^*}{\vdash \Phi : \Phi'', \Xi_2, D(G)[h] \Downarrow D^\perp(H_1)[f']} \ (R \Downarrow)$$

Here $g : V_{H_1} \setminus \text{ran}(ext_{H_1}) \to \text{Var}$ is a substitution such that, for each $v \in V_{H_1} \setminus \text{ran}(ext_{H_1})$, $g(v)$ is a fresh variable, i.e. one not occuring in $\Xi_2, D(G)[h], \mathcal{D}^\perp(H_1)[f']$ (for $v \in \text{dom}(g)$).

Let ξ be a variable. We shall prove now that $\nu(\xi)$ cannot occur twice in $\Xi_2, \mathcal{D}^\perp(H_1)[f'][g]$. The formula $\nu^\perp(\xi)$ cannot occur in Ξ_2 twice because it occurs in Ξ at most once. It cannot occur in $\mathcal{D}^\perp(H_1)[f'][g]$ twice as well. Indeed, no formula of the form $\nu^\perp(x)$ occurs twice in $\mathcal{D}^\perp(H_1)$ (see Definition 18). The functions f and χ_p are injective, so the same property holds for $\mathcal{D}^\perp(H_1)[f']$. Finally, the function g is also injective because it replaces some variables in $\mathcal{D}^\perp(H_1)[f']$ by fresh ones. Therefore, the desired property holds for the multiset $\mathcal{D}^\perp(H_1)[f'][g]$ as well.

Finally, assume, ex falso, that $\nu^\perp(\xi)$ occurs once in Ξ_2 and once in $\mathcal{D}^\perp(H_1)[f'][g]$. It is not the case that $\xi \in \text{ran}(g)$ because g introduces only fresh variables. Therefore, $\xi = f'(v) = f(\chi_p(v))$ for $v = ext_{H_1}(\sigma)$, $\sigma \in \text{type}(H_1)$. Consequently, $\xi = f(ext_{H_2}(\sigma)) = ext_{H_2'}(\sigma)$. However, $\nu^\perp(ext_{H_2'}(\sigma))$ belongs to $\Xi_1 = \mathcal{D}^\perp(H_2')$, so it occurs in $\Xi = \Xi_1, \Xi_2$ twice. This is a contradiction.

Summing up, one can apply the induction hypothesis to the sequent

$$\vdash \Phi : \Phi'', \Xi_2, \mathcal{D}^\perp(H_1)[f'][g], D(G)[h] \Uparrow$$

and conclude that

- h is injective;
- $\Xi_2, \mathcal{D}^\perp(H_1)[f'][g] = \mathcal{D}^\perp(H')$ for a hypergraph H' such that $\text{type}(H') = \text{type}(G)$, $h(ext_G(\sigma)) = ext_{H'}(\sigma)$ for $\sigma \in \text{type}(G)$;
- there is a derivation of a hypergraph isomorphic to H' from G that uses each rule from P'' exactly once and that can use rules from P arbitrary many times.

Since $g \circ f'$ is injective, it holds that $\mathcal{D}^{\perp}(H_1)[f'][g] = \mathcal{D}^{\perp}(H_1')$ where H_1' is isomorphic to H_1. Let K be a hypergraph obtained from H' by removing nodes belonging to $\mathrm{ran}(g)$, removing hyperedges belonging to $E_{H_1'}$, and adding a new hyperedge e_0 such that $\mathrm{type}_K(e_0) = \mathrm{type}(H_1)$ and $att_K(e_0)(\sigma) = ext_{H_1'}(\sigma)$. Clearly, $K[e_0/H_1]$ is isomorphic to H'.

The multiset Ξ_2 equals

$$\Xi_2 = \{lab_K(e)[att_K(e)] \mid e \in E_K \setminus \{e_0\}\} \cup \{\nu(v) \mid v \in V_K \setminus \mathrm{ran}(att_K(e_0))\}.$$

Earlier, we proved that $\Xi_1 = \mathcal{D}^{\perp}(H_2')$ for H_2' being isomorphic to H_2. For $\sigma \in \mathrm{type}(H_2')$, it holds that $ext_{H_2'}(\sigma) = f(ext_{H_2}(\sigma)) = f(\chi_p(ext_{H_1}(\sigma))) = f'(ext_{H_1}(\sigma)) = ext_{H_1'}(\sigma)$.

Assume that $v \in V_{H_2'} \setminus \mathrm{ran}(ext_{H_2'})$. Then, $\nu(v)$ belongs to Ξ_1. If v appears as a free variable in Ξ_2, then, since $v \notin \mathrm{ran}(ext_{H_2'}) = \mathrm{ran}(att_K(e_0))$, there must be the formula $\nu(v)$ in Ξ_2. But this would imply that $\nu(v)$ occurs twice in Ξ, which is a contradiction. Therefore, v does not appear in Ξ_2. All this implies that $\Xi_1, \Xi_2 = \mathcal{D}^{\perp}(G')$ where G' is isomorphic to $K[e_0/H_2'] = K[e_0/H_2]$. Summing up, G' is derivable from G using rules from P, using exactly once rules from P'' and the rule $p = (H_1 \to H_2)$.

The only thing remaining to be proved is that $h(ext_G(\sigma)) = ext_{G'}(\sigma)$. We know that $h(ext_G(\sigma)) = ext_{H'}(\sigma)$. The procedure described above that transforms H' into G' does not affect external nodes. Therefore, $ext_{H'}(\sigma) = ext_{G'}(\sigma)$, as desired.

Case 3.

$$\frac{\vdash \Phi : \Phi', \Xi, D(G)[h] \Downarrow fm^{\perp}(p)}{\vdash \Phi : \Phi', \Xi, D(G)[h] \Uparrow} \ (D_1)$$

Here $fm^{\perp}(p) \in \Phi$. This case is analogous to Case 2.

Now, let us proceed to the second statement of the Lemma. Assume that the sequent $\vdash \{fm^{\perp}(r) \mid r \in P\} : \Theta \Downarrow D(G)[h]$ is derivable. The formula $D(G)[h]$ is of the form $\exists \vec{v} \bigotimes \mathcal{D}(G)[h]$; thus, any derivation of the sequent must end as follows:

$$\frac{\vdash \{fm^{\perp}(r) \mid r \in P\} : \Theta \Downarrow \bigotimes \mathcal{D}(G)[h][g]}{\vdash \{fm^{\perp}(r) \mid r \in P\} : \Theta \Downarrow D(G)[h]} \ (\exists)^*$$

Here g maps nodes from $V_G \setminus \mathrm{ran}(ext_G)$ to variables. The premise can only be obtained by consecutive applications of $(\otimes)$. Going in this proof from bottom to top, one finally reaches sequents of the form $\vdash \{fm^{\perp}(r) \mid r \in P\} : \tilde{\Theta} \Downarrow X$ where $\tilde{\Theta}$ is a subset of Θ and X is an atom from $\mathcal{D}(G)[h][g]$. Such a sequent can only be an axiom, so $\tilde{\Theta} = X^{\perp}$. Going back from top to bottom, we conclude that $\Theta = \mathcal{D}^{\perp}(G)[h][g]$.

The functions h and g have disjoint domains such that $\mathrm{dom}(h) \cup \mathrm{dom}(g) = V_G$. Let us unify these functions into a single one f. If f is not injective, e.g. $f(v_1) = f(v_2) = x$, then Θ contains $\nu(x)$ twice, which is a contradiction. Thus, f is injective. This implies that h is injective and that $\Theta = \mathcal{D}^{\perp}(G')$ where G' is isomorphic to G via the node isomorphism f; besides, $h(ext_G(\sigma)) = ext_{G'}(\sigma)$.

The above reasonings, in fact, also prove the "if" parts of Lemma 37, because they show how to construct derivations of sequents of interest.   ◀

Lemma 22 follows directly from Lemma 37. Indeed, the sequent

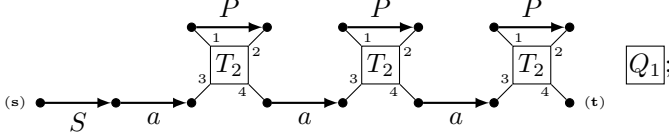$$\{!fm(r) \mid r \in P\}, \{fm(r) \mid r \in P'\}, \mathcal{D}(G') \vdash D(G)[\chi_{G \to G'}]$$
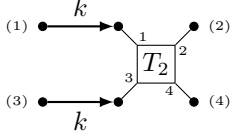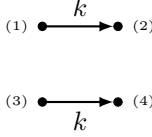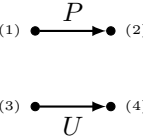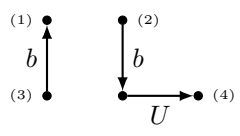
is derivable in ILL1 if and only if its translation into LL1

$$\vdash \{fm^{\perp}(r) \mid r \in P\} : \{fm^{\perp}(r) \mid r \in P'\}, \mathcal{D}^{\perp}(G'), D(G)[\chi_{G \to G'}] \Uparrow$$

is derivable in Σ_3 [1, Theorems 1, 2]. This, according to Lemma 37, is equivalent to the fact that a hypergraph isomorphic to G' is derivable from G using each rule from P' once and using some rules from P.

A.3 Alternative Proof of Theorem 30

In view of Corollary 28, it suffices to present a linear-time ht-system generating an NP-complete language of string graphs. Let $P, Q_1, Q_2, S, T_1, T_2, U$ be nonterminal symbols such that $\text{type}(Q_1) = \text{type}(Q_2) = \emptyset$, $\text{type}(P) = \text{type}(S) = \text{type}(T_1) = \text{type}(U) = \{\mathbf{s}, \mathbf{t}\}$ and $\text{type}(T_2) = \{1, 2, 3, 4\}$; let $0, 1, a, b, c$ be terminal symbols with type $\{\mathbf{s}, \mathbf{t}\}$. Let the start hypergraph be $S^\bullet + Q_1^\bullet$ ('+' is introduced in Definition 5). The rules are presented below.

1. $S^\bullet + Q_1^\bullet \to \text{sg}(SaT_1aT_1aT_1) + Q_1^\bullet$;

2. $T_1^\bullet + Q_1^\bullet \to \text{sg}(kT_1) + Q_1^\bullet$, $\quad T_1^\bullet + Q_1^\bullet \to \text{sg}(k) + Q_1^\bullet$ for $k = 0, 1$;

3. $S^\bullet + Q_1^\bullet \to$

4. $T_2^\bullet + Q_1^\bullet \to$ for $k = 0, 1$;

5. $S^\bullet + Q_1^\bullet \to U^\bullet + Q_2^\bullet$;

6.

7. $U^\bullet + Q_2^\bullet \to c^\bullet$.

This ht-system is linear-time because each production, except for 5, which is applied at most once in any derivation, increases the number of terminal symbols. This ht-system generates hypergraphs of the form

$$\text{sg}(bw_1 bb w_2 b \ldots bw_m bcax_1 ay_1 az_1 \ldots ax_n ay_n az_n) \tag{4}$$

such that w_i, x_i, y_i, z_i are nonempty strings over the alphabet $\{0, 1\}$ and such that $\{w_1, \ldots, w_m\}$ coincides as a multiset with $\{x_{i_1}, y_{i_1}, z_{i_1}, \ldots, x_{i_l}, y_{i_l}, z_{i_l}\}$ for some $1 \le i_1 < \ldots < i_l \le n$. Consequently, one can reduce the exact cover problem by 3-sets to this language: if $X = \{w_1, \ldots, w_m\}$ is a set and $C = \{\{x_1, y_1, z_1\}, \ldots, \{x_n, y_n, z_n\}\}$ is a collection of 3-element subsets of X, then checking whether X is the disjoint union of some sets from C is equivalent to checking whether the hypergraph (4) is generated by the ht-system.

A.4 Proof of Theorem 35

▶ **Lemma 38.**

1. *Hypergraphs* $H_1 = H_{\Gamma; A_1}^{X; Y_1}$ *and* $H_2 = H_{\Gamma; A_2}^{X; Y_2}$ *are equal (as concrete hypergraphs) if and only if* $\text{type}(H_1) = \text{type}(H_2)$.

2. *If* v *is an isolated node in* $H = H_{\Gamma; A}^{X; Y}$, *then* $H' = H_{\Gamma; A}^{X \setminus \{v\}; Y \setminus \{v\}}$ *is obtained from* H *by removing* v.

Proof. 1. This follows from the definition of $H = H_{\Gamma;A}^{X;Y}$, because Y, A define only the domain of ext_H.

2. The node v being isolated is equivalent to the fact that $v \notin \mathrm{FVar}(\Gamma)$. Therefore, $V_{H'} = \mathrm{FVar}(\Gamma) \cup (X \setminus \{v\}) = (\mathrm{FVar}(\Gamma) \cup X) \setminus \{v\} = V_H \setminus \{v\}$. Clearly, only the node v is affected, so the statement follows.

$\blacktriangleleft$

We start with showing that $sub_h(\mathrm{u}(A)) \subseteq \mathrm{u}(A[h])$. Clearly, it suffices to check this property only for the case where h changes only one variable, e.g. $h(z) = t$ ($z \neq t$) and $h(x) = x$ for $x \neq z$. Indeed, $sub_h(sub_g(H)) = sub_{h \circ g}(H)$ and sub_h is a monotone operation.

Let $H = H_{\Gamma;A}^{X;Y}$ for $\Gamma \vdash A$ being derivable and let $h(z) = t$ ($z \neq t$) and $h(x) = x$ for $x \neq z$. Our goal is to show that a hypergraph isomorphic to $sub_h(H)$ belongs to $\mathrm{u}(A[h])$.

Case 1. $z \notin \mathrm{type}(H)$. Then $sub_h(H) = H$. If $z \notin \mathrm{FVar}(A)$, then $A[t/z] = A$, so $H \in \mathrm{u}(A[t/z])$, and we are done.

Assume that $z \in \mathrm{FVar}(A)$; then, $z \notin (\mathrm{FVar}(\Gamma) \cup X)$. Consequently, $\Gamma \vdash A[t/z]$ is derivable because z does not occur freely in Γ. Let us define $H' := H_{\Gamma;A[t/z]}^{X;Y} \in \mathrm{u}(A[t/z])$. H' and H can possibly differ only in $\mathrm{type}(H)$ and $\mathrm{type}(H')$; if $\mathrm{type}(H) = \mathrm{type}(H')$, then $H = H'$ (Lemma 38), and the proof is completed. Since $z \notin \mathrm{type}(H)$, $z \notin \mathrm{type}(H')$. Besides, if $t \in \mathrm{type}(H)$, then $t \in \mathrm{type}(H')$.

Assume that $t \in \mathrm{type}(H') \setminus \mathrm{type}(H)$ (this is the only remaining possibility for $\mathrm{type}(H)$ and $\mathrm{type}(H')$ to differ). This implies that $t \notin Y$ and that $t \notin (\mathrm{FVar}(\Gamma) \cup X) \cap \mathrm{FVar}(A)$. Let τ be a fresh variable not occuring in Γ, A, X, Y. Then, $\hat{H} := H_{\Gamma[\tau/t];A}^{X[\tau/t];Y}$ is isomorphic to H and $\Gamma[\tau/t] \vdash A$ is derivable. Indeed, either t is not in $\mathrm{FVar}(\Gamma) \cup X$, and hence substitution changes nothing; or t is not free in A, hence $\hat{H}$ is obtained from H by just renaming the node t as τ. External nodes are not changed because t is not external in H and the new node τ is not external in $\hat{H}$ because it does not belong to $\mathrm{FVar}(A)$ or to Y.

The sequent $\Gamma[\tau/t] \vdash A[t/z]$ is derivable because z does not occur freely in $\Gamma[\tau/t]$. Consider $\hat{H}' = H_{\Gamma[\tau/t];A[t/z]}^{X[\tau/t];Y}$. It holds that $t \notin \mathrm{type}(\hat{H}')$ as well as $t \notin \mathrm{type}(\hat{H})$. Thus, $\mathrm{type}(\hat{H}) = \mathrm{type}(\hat{H}')$, so $\hat{H} = \hat{H}' \in \mathrm{u}(A[t/z])$. Therefore, H belongs to $\mathrm{u}(A[t/z])$, as we consider membership to a hypergraph language up to isomorphism.

Case 2. $z \in \mathrm{type}(H)$, $t \notin \mathrm{type}(H)$. Let τ be a fresh variable; define a function g such that $g(z) = t$, $g(t) = \tau$, otherwise identical. The hypergraph H is isomorphic to $H_1 := H_{\Gamma[\tau/t];A[\tau/t]}^{X[\tau/t];Y[\tau/t]}$. Indeed, H_1 is obtained from H by simply renaming t by τ and, since t is not external, external nodes of H remain unchanged. Now, note that $sub_h(H_1)$ is isomorphic to $H_2 := H_{\Gamma[g];A[g]}^{X[g];Y[g]}$. Indeed, $sub_h(H_1)$ is a renaming of the external node z by t; in turn, H_2 is obtained from H_1 by changing the node z to t (with $ext_{H_2}(t) = t$).

Finally, H_2 equals $H_3 := H_{\Gamma[g];A[t/z]}^{X[g];Y[g]}$. This can be verified by Lemma 38. Indeed, both $\mathrm{type}(H_2)$ and $\mathrm{type}(H_3)$ do not contain z because z does not belong even to V_{H_2} and to V_{H_3}. Besides, $t \in \mathrm{type}(H_2)$ and $t \in \mathrm{type}(H_3)$ because $z \in \mathrm{type}(H_1)$; finally, $\tau \notin \mathrm{type}(H_2)$. Thus, $\mathrm{type}(H_2) = \mathrm{type}(H_3)$, hence $H_2 = H_3 \in \mathrm{u}(A[t/z])$. We have proved that a hypergraph isomorphic to $sub_h(H)$ belongs to $\mathrm{u}(A[t/z])$, as desired.

Case 3. $z, t \in \mathrm{type}(H)$. In this case, take $H' := H_{\Gamma[h];A[h]}^{X[h];Y[h]}$. H' is obtained from H by identifying the node z with the node t, the resulting node being named t. Thus, clearly, H' is isomorphic to $sub_h(H)$.

Let us check that $\mathrm{u}(\forall x A) = \bigcap_{y \in \mathrm{Var}} \mathrm{u}(A[y/x])$. Take $H = H_{\Gamma;\forall x A}^{X;Y} \in \mathrm{u}(\forall x A)$.

- If $y \notin V_H = \mathrm{FVar}(\Gamma) \cup X$, then $H_{\Gamma;\forall x A}^{X;Y} = H_{\Gamma;A[y/x]}^{X;Y}$, and, since $\Gamma \vdash A[y/x]$ is derivable, it holds that $H \in \mathrm{u}(A[y/x])$.

- If $y \in \mathrm{ran}(ext_H) = (V_H \cap \mathrm{FVar}(\forall x A)) \cup Y$, then it is also the case that $H_{\Gamma;\forall x A}^{X;Y} = H_{\Gamma;A[y/x]}^{X;Y}$ and we are done.
- Let $y \in V_H \setminus (\mathrm{FVar}(\forall x A) \cup Y)$. Let z be a fresh variable and let Γ' and X' be obtained by replacing y by z in Γ and X resp. Then, $H = H_{\Gamma;\forall x A}^{X;Y}$ is isomorphic to $H_{\Gamma';\forall x A}^{X';Y}$ and $\Gamma' \vdash \forall x A$ is derivable (since z is not free in $\forall x A$); in turn, $H_{\Gamma';\forall x A}^{X';Y} = H_{\Gamma';A[y/x]}^{X';Y}$ because $y \notin \mathrm{FVar}(\Gamma') \cup X'$. This implies that H (as an abstract hypergraph) belongs to $\mathrm{u}(A[y/x])$, as desired. Note that $\Gamma' \vdash A[y/x]$ is derivable because of invertibility of $(\forall R)$.

Thus, $\mathrm{u}(\forall x A) \subseteq \bigcap_{y \in \mathrm{Var}} \mathrm{u}(A[y/x])$. Let us show the converse inclusion. Let $H \in \bigcap_{y \in \mathrm{Var}} \mathrm{u}(A[y/x])$ and let z be a fresh variable not in $\mathrm{type}(H)$. Then, for some X, Y, Γ, it holds that $H = H_{\Gamma;A[z/x]}^{X;Y}$ and that $\Gamma \vdash A[z/x]$ is derivable. Note that $\mathrm{type}(H) = ((\mathrm{FVar}(\Gamma) \cup X) \cap \mathrm{FVar}(A[z/x])) \cup Y$; therefore, $z \notin \mathrm{FVar}(\Gamma) \cup X$ and, by $(\forall R)$, the sequent $\Gamma \vdash \forall x A$ is derivable. It remains to observe that $H_{\Gamma;A[z/x]}^{X;Y} = H_{\Gamma;\forall x A}^{X;Y}$.

Let us check that $\mathrm{u}(A \multimap B) \subseteq \{H \mid \forall H' \in \mathrm{u}(A)\, (H \mathbin{/\!/} H' \in \mathrm{u}(B))\}$. Take hypergraphs $H_1 = H_{\Gamma;A \multimap B}^{X;Y} \in \mathrm{u}(A \multimap B)$ and $H_2 = H_{\Delta;A}^{Z;W} \in \mathrm{u}(A)$. Without loss of generality, let us assume that variables in $(\mathrm{FVar}(\Gamma) \cup X) \setminus \mathrm{type}(H_1)$ and $(\mathrm{FVar}(\Delta) \cup Z) \setminus \mathrm{type}(H_2)$ are replaced by fresh ones (such a replacement results in isomorphic hypergraphs). Nodes in H_1 are variables from $\mathrm{FVar}(\Gamma) \cup X$ and nodes in H_2 are variables from $\mathrm{FVar}(\Delta) \cup Z$.

Let $H_3 := H_{\Gamma,\Delta;B}^{X \cup Z;Y \cup W \cup U}$ where $U = (\mathrm{FVar}(\Gamma, \Delta) \cup X \cup Z) \cap \mathrm{FVar}(A)$. We claim that $H_1 \mathbin{/\!/} H_2 = H_3$. First, note that the above variable freshness condition guarantees that only common external nodes of H_1 and H_2 are identified in H_3. Let us also check that external nodes in $H_1 \mathbin{/\!/} H_2$ and in H_3 are the same, i.e. that $\mathrm{type}(H_1 \mathbin{/\!/} H_2) = \mathrm{type}(H_3)$.

$$
\begin{aligned}
\mathrm{type}(H_1 \mathbin{/\!/} H_2) &= \mathrm{type}(H_1) \cup \mathrm{type}(H_2) \\
&= ((\mathrm{FVar}(\Gamma) \cup X) \cap \mathrm{FVar}(A \multimap B)) \cup Y \cup \\
&\qquad ((\mathrm{FVar}(\Delta) \cup Z) \cap \mathrm{FVar}(A)) \cup W \\
&\subseteq ((\mathrm{FVar}(\Gamma, \Delta) \cup X \cup Z) \cap \mathrm{FVar}(B)) \cup Y \cup W \cup U = \mathrm{type}(H_3).
\end{aligned}
$$

To check that $\mathrm{type}(H_1 \mathbin{/\!/} H_2) \supseteq \mathrm{type}(H_3)$, assume that $x \in \mathrm{type}(H_3) \setminus (\mathrm{type}(H_1) \cup \mathrm{type}(H_2))$. This implies that x belongs to $(\mathrm{FVar}(\Delta) \cup Z) \cap \mathrm{FVar}(B)$. However, this contradicts the fact that x is fresh for B. Thus, $\mathrm{type}(H_3) = \mathrm{type}(H_1) \mathbin{/\!/} \mathrm{type}(H_2)$.

To show that $\mathrm{u}(A \multimap B) \supseteq \{H \mid \forall H' \in \mathrm{u}(A)\, (H \mathbin{/\!/} H' \in \mathrm{u}(B))\}$, assume that we are given a hypergraph H such that $H \mathbin{/\!/} H'$ belongs to $\mathrm{u}(B)$ for each $H' \in \mathrm{u}(A)$. Take $H' := H_{A;A}^{\mathrm{type}(H);\mathrm{type}(H)} \in \mathrm{u}(A)$. Then $H \mathbin{/\!/} H' = H_{\Gamma;B}^{X;Y}$ for some X, Y, Γ. Clearly,

- $\Gamma = A, \Gamma'$;
- $\mathrm{type}(H \mathbin{/\!/} H') = \mathrm{type}(H) \cup \mathrm{FVar}(A)$;
- H is obtained from $H_{\Gamma;B}^{X;Y}$ by removing one A-labeled hyperedge and then external nodes from $\mathrm{FVar}(A) \setminus \mathrm{type}(H)$. To be more precise, if $x \in \mathrm{FVar}(A) \setminus \mathrm{type}(H)$, then $x \in \mathrm{type}(H \mathbin{/\!/} H')$ but $x \notin \mathrm{type}(H)$; thus, after removing the A-labeled hyperedge from $H_{\Gamma;B}^{X;Y}$, the node x becomes isolated. Thus, $x \notin \mathrm{FVar}(\Gamma')$ because otherwise x would be an attachment node of a hyperedge corresponding to a formula from Γ'.

First, consider the hypergraph $H_1 = H_{\Gamma';B}^{X \cup \mathrm{FVar}(A);Y}$. It is obtained from $H_0 = H_{\Gamma;B}^{X;Y}$ by removing an A-labeled hyperedge and changing nothing else. Nodes from $\mathrm{FVar}(A) \setminus \mathrm{type}(H)$ are isolated in H_1, and $\mathrm{type}(H_1) = \mathrm{type}(H_0)$. Secondly, we claim that H_1 is equal to $H_2 = H_{\Gamma';A \multimap B}^{X \cup \mathrm{FVar}(A);Y}$. Indeed, $\mathrm{type}(H_2) \setminus \mathrm{type}(H_1) \subseteq (\mathrm{FVar}(\Gamma') \cup X \cup \mathrm{FVar}(A)) \cap \mathrm{FVar}(A) = \mathrm{FVar}(A)$, but $\mathrm{type}(H_1) = \mathrm{type}(H_0) \supseteq \mathrm{FVar}(A)$, so $\mathrm{type}(H_2) \setminus \mathrm{type}(H_1) = \emptyset$ and thus $\mathrm{type}(H_1) = \mathrm{type}(H_2)$. Applying Lemma 38 shows that $H_1 = H_2$.

Finally, the second statement of Lemma 38 implies that there are X', Y' such that $H_{\Gamma';A \multimap B}^{X';Y'}$ is obtained from H_2 by removing nodes that belong to $\mathrm{FVar}(A) \setminus \mathrm{type}(H)$. Therefore,

$H_{\Gamma';A\multimap B}^{X';Y'}$ is isomorphic to H and thus $H \in \mathrm{u}(A \multimap B)$.

Summing up, we have proved that $\langle \mathrm{T}, \mathrm{u} \rangle$ is a hypergraph language model.