A Lambek Automaton

Tatyana Veksler, *Computer Science department, Technion, Haifa, email: tatyana@il.ibm.com*

Nissim Francez, *Computer Science department, Technion, Haifa, email: francez@cs.technion.ac.il*

Abstract

We define an automata-theoretic counterpart of (type-logical) grammars based on the (associative) *Lambek-calculus* **L**, a prominent formalism in computational linguistics. While the usual push-down automaton (PDA) has the same *weak generative power* as the **L**-based grammars (Pentus, 1995), there is no direct relationship between the computations of a PDA for some language L and the derivations of an **L**-based grammar for L. In the Lambek-automaton, on the other hand, there is a *tight relation* (1-1) between automaton computations and grammar derivations. The automaton exhibits a novel mode of operation, using *hypothetical* steps, directly inspired by the hypothetical reasoning embodied by **L**.

Keywords: Lambek-Automaton, Lambek-calculus, formal languages, type-logical grammar

1 Introduction

Type-Logical Grammars (TLG) [3] constitute a major formalism in computational linguistics for specifying the syntax and semantics of natural languages, as well as formal languages. We concentrate here on the grammars based on the associative Lambek-calculus **L** [2]. This calculus underlies many *TLG* grammars, and has a very strong explanatory power regarding the syntax-semantics interface, as well as presenting a radically different approach to formal languages theory. Much of the research of **L**-grammars has been devoted to locating their *generative capacity* in Chomsky hierarchy [1](Section 3.2). Since the 60s, the main open problem in this field has been the 'Chomsky Conjecture', stating that the languages recognizable by **L**-grammars are precisely the context-free ones. This was finally proved by Pentus [4].

A natural question associated with every kind of grammar is, what is its *automata-theoretic counterpart*? In view of Pentus' result, an obvious possibility is the standard pushdown automaton (PDA), standardly associated with context-free languages. Indeed, if *weak-generative capacity* is the issue, PDAs will do. However, in analogy to *strong-generative capacity*, one might be interested in an automaton, that not only accepts the same family of languages, but does so in a way *tightly-related* (a term to be made precise in the sequel) to the way the grammars operate. The counterpart of context-free grammars (CFGs), inherently connected with *recursion*, is indeed the PDA[1]. However, PDAs are not directly related to the *way* **L**-grammars generate a language; in particular, they do not reflect in any way *hypothetical reasoning*, a major ingredient in **L**, allowing assumptions that are *discharged* during derivation

[1] Though, by standard constructions, PDAs are tightly-related only to CFGs in Greibach normal form.

(deduction!).

In this paper we propose an automata-theoretic counterpart of **L**-grammars, namely the **L**-automaton, which we consider more adequate due to its *tight-relatedness* to **L**-grammars. Roughly, this means a 1-1 *simulation* relation between derivations of an **L**-grammar and computations of its associated automaton. The automaton accepts a string "in a similar way" to the grammar derivation. Another view (of the same) is the availability of a construction of an automaton M_G for every **L**-grammar G, and a construction of a grammar G_M for every **L**-automaton M, such that

$$(*) M_{G_M} \approx M, \quad G_{M_G} \approx G$$

In section 2 we give a brief overview of the Lambek calculus **L** and its use for specifying the grammars, also setting the notation. In section 3 we define the **L**-automaton and investigate some of its properties. In section 4 we propose constructions $M \rightsquigarrow G_M$, $G \rightsquigarrow M_G$, satisfying (*) above. In section 5 we summarize the results and suggest possible continual research.

2 Preliminaries

2.1 Categories

The point of departure is some given finite set $\mathcal{B} \neq \emptyset$ of *basic categories*. Uppercase roman letters $A, B, C, \ldots$ denote basic categories. This set is further closed to form complex categories from simpler ones by means of *directed arrows*[2].

DEFINITION 2.1
The set of **categories** $\mathcal{C}$ is the least set such that:

- $\mathcal{B} \subseteq \mathcal{C}$
- if $\tau_1, \tau_2 \in \mathcal{C}$, then $(\tau_1 \to \tau_2), (\tau_2 \leftarrow \tau_1) \in \mathcal{C}$

Meta-variables τ, τ_i range over categories. An informal explanation of the complex category meaning is as follows: if a string u of a category τ_1 is concatenated to the *left* of a string w having a complex category $(\tau_1 \to \tau_2)$, then the resulting string uw is of a category τ_2. The meaning of left arrow complex categories is similar, but with concatenation to the *right*.

DEFINITION 2.2
The **length of a category** τ, $\#(\tau)$ is defined recursively:

- $\#(A) = 1$ if $A \in \mathcal{B}$
- $\#(\tau_2 \leftarrow \tau_1) = \#(\tau_1 \to \tau_2) = \#(\tau_1) + \#(\tau_2)$

DEFINITION 2.3
For $\mathcal{A} \subseteq \mathcal{C}$, its **subcategory-closure** $SC(\mathcal{A})$ is the least set such that

1. $\mathcal{A} \subseteq SC(\mathcal{A})$, and
2. If $(\tau_2 \leftarrow \tau_1) \in SC(\mathcal{A})$, then $\tau_1 \in SC(\mathcal{A})$ and $\tau_2 \in SC(\mathcal{A})$, and
3. If $(\tau_1 \to \tau_2) \in ST(\mathcal{A})$, then $\tau_1 \in ST(\mathcal{A})$ and $\tau_2 \in ST(\mathcal{A})$.

[2] We prefer the more readable 'arrow notation' over the traditional 'slash notation'

$$\tau \rhd \tau \ \textbf{(ax)}$$

$$\frac{\Gamma_1 \rhd \tau_1 \quad \Gamma_2 \rhd (\tau_1 \to \tau_2)}{\Gamma_1\Gamma_2 \rhd \tau_2} \ (\to \boldsymbol{E}) \qquad \frac{\Gamma_2 \rhd (\tau_2 \leftarrow \tau_1) \quad \Gamma_1 \rhd \tau_1}{\Gamma_2\Gamma_1 \rhd \tau_2} \ (\leftarrow \boldsymbol{E})$$

$$\frac{\tau_1\Gamma \rhd \tau_2}{\Gamma \rhd (\tau_1 \to \tau_2)} \ (\to \boldsymbol{I}) \qquad \frac{\Gamma\tau_1 \rhd \tau_2}{\Gamma \rhd (\tau_2 \leftarrow \tau_1)} \ (\leftarrow \boldsymbol{I}) \quad \Gamma \neq \epsilon$$

FIG. 1. The Natural Deduction style formalization of **L**

2.2 **L**-*grammars*

DEFINITION 2.4
A **type-logical (or Categorial) grammar** is a tuple $G = (\Sigma, \mathcal{B}, \tau_0, \alpha)$, where:

- Σ is a finite alphabet
- $\mathcal{B}$ is a finite set of basic categories (with $\mathcal{C}$ the induced categories)
- $\tau_0 \in \mathcal{C}$ is the initial category
- $\alpha : \Sigma \longrightarrow \mathcal{P}_{fin}(\mathcal{C})$ is a mapping assigning to each terminal symbol a finite subset of $\mathcal{C}$. By convention, $\alpha[\sigma] \neq \emptyset$ for all $\sigma \in \Sigma$.

We lift α to an assignment to $w \in \Sigma^+$ by letting, for $w = \sigma_1 \dots \sigma_n$, $\alpha[w] = \alpha[\sigma_1] \cdot \ldots \cdot \alpha[\sigma_n]$. Note, that this is in general a set of sequences.

A categorial grammar is based on a *deductive system*, (a *syntactic calculus*) determining how a sequence of categories is *reduced* to a single category. A concatenation of categories τ_1, τ_2 is denoted by $\tau_1\tau_2$. We use meta-variables Γ, Γ_i to range over finite sequences of categories. Concatenation of such sequences Γ_1 and Γ_2 is denoted by $\Gamma_1\Gamma_2$. For $\Gamma = \tau_1 \dots \tau_n$ a non-empty sequence of categories and τ a category, the expression (sequent) $\Gamma \rhd \tau$ is called a *reducibility statement*[3], and means that the sequence $\tau_1 \dots \tau_n$ is *reducible* to τ. The reducibility relation depends on the underlying logic. For a logic L, $\vdash_L \Gamma \rhd \tau$ denotes *provability* in L of the sequent. Here, we are mainly considered with **L**, presented in Figure 1 in its *natural deduction* presentation, as the underlying syntactic calculus. Let G be an **L**-grammar, i.e., a grammar based on, in the above sense, on **L** as its underlying deduction system. The (formal) *language generated* by G is

DEFINITION 2.5

$$L(G) \stackrel{\triangle}{=} \{w \in \Sigma^+ \mid \exists\, \Gamma \in \alpha[w] \text{ such that } \vdash_{\mathbf{L}} \Gamma \rhd \tau_0\}$$

[3]Known also as a *sequent*. Note that in the absence of any *structural rules* (besides associativity), the antecedent Γ of a sequent $\Gamma \vdash \tau$ is a sequence of categories, in contrast to being a (multi-)set in classical/intuitionistic logic.

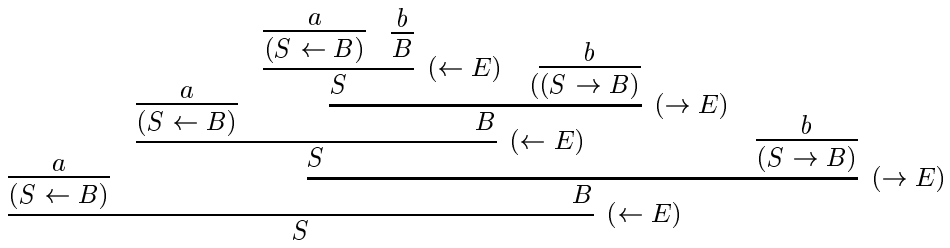


FIG. 2. An **AB**-derivation of a^3b^3

In words, $L(G)$ consists of all strings in Σ^+ for which there is a lexical category-selection, such that the reducibility of the sequence of selected categories to the distinguished category is provable in **L**. When $w \in L[\![G]\!]$ due to $\vdash_{\mathbf{L}} \alpha[\![w]\!] = \Gamma \rhd \tau_0\}$, we refer to a derivation of $\Gamma \rhd \tau_0$ as a *witness-derivation* for w. Throughout the paper, we relate only to *normal* **L**-derivations (where it is well-known that **L** strongly-normalizesa.)

The sublogic without arrow-introduction rules is known as the **AB**-calculus, or *applicative* syntactic calculus. In the arrow elimination rules the premise containing the eliminated arrow is the *major premise*, and the other premise is the *minor premise*. The two arrow-introduction rules $(\rightarrow I)$ and $(\leftarrow I)$ embody the *hypothetical reasoning* capacity of **L**. The main idea of hypothetical reasoning is temporarily *assuming* a hypothesis A, using A to prove some statement B and, eventually, *discharging* the hypothesis A, while deducing that A implies B. In **L**, only *peripheral* (i.e., either *leftmost* or *rightmost* elements in Γ) assumptions may be discharged via the arrow introduction rules. The non-emptyness side-condition on Γ in the arrow introduction-rules originates from the use of **L** for grammars, not from any logical reason. In an **L**-based grammar, Γ corresponds to a sequence of categories assigned to some string $w \in \Sigma^+$ (and not Σ^*), as ϵ (the empty string) cannot be *lexically* interpreted.

Derivation trees are finite binary trees, which are presented with the leaves on top and the root at the bottom. Every node in the tree is labeled with a sequent (or, in a short notation, with a formula which is the right-hand side of a sequent). Every leaf is labeled with an axiom and, going from the leaves to the root, the label of a node is obtained from the labels of the nodes right above it by an application of an inference rule . Leaves representing assumptions discharged somewhere in the derivation are marked with square brackets. A sequent $\Gamma \rhd \tau$ ($\Gamma = \tau_1 \ldots \tau_n$) is *derivable* in **L** if there is a derivation tree in **L** with the leaves labeled by $\tau_1 \rhd \tau_1, \ldots, \tau_n \rhd \tau_n$ (or by $\tau_1, \ldots \tau_n$ in a short notation) and the root labeled by $\Gamma \rhd \tau$ (or by τ in a short notation). For convenience, we use the short notation in examples.

EXAMPLE 2.6
An **AB**-grammar generating the context-free language $\{a^n b^n \mid n > 0\}$:
$G = (\{a, b\}, \{B, S\}, S, \alpha)$, where $\alpha[a] = \{(S \leftarrow B)\}$, $\alpha[b] = \{B, (S \rightarrow B)\}$.
A derivation of a^3b^3 is shown in Figure 2.

$$\dfrac{\dfrac{\dfrac{[(A \leftarrow (A \to A))]}{A}\,(\leftarrow E)}{((A \leftarrow (A \to A)) \to A)}\,(\to I) \qquad \dfrac{\dfrac{[A]\ [(A \to A)]}{A}\,(\to E)}{(A \leftarrow (A \to A))}\,(\leftarrow I)}{A}\,(\to E)$$

FIG. 3. Ambiguity in **L**

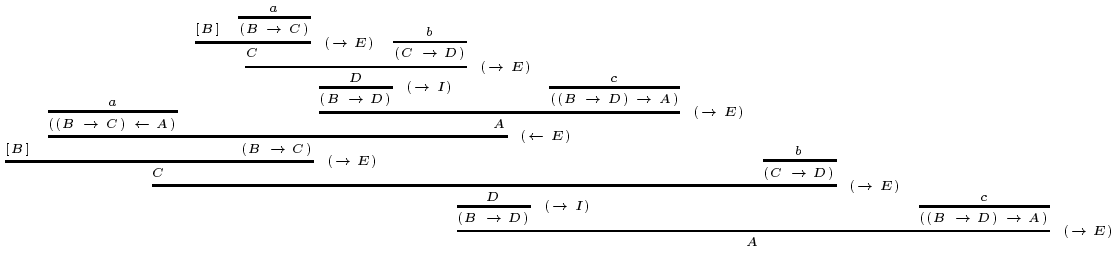


FIG. 4. An **L**-derivation of *aabcbc*

A sequent may have more than one **L**-derivation. For example, the sequent $\tau \rhd \tau$, where $\tau = ((A \leftarrow (A \to A)) \to A)$, has two derivations: one is simply the axiom $\tau \rhd \tau$, and the other one is shown in Figure 3.

Following are two examples of **L**-grammars.

EXAMPLE 2.7
$G_1 = (\{a, b, c\}, \{A, B, C, D\}, A, \alpha)$,
$\alpha[a] = \{((B \to C) \leftarrow A), (B \to C)\}$, $\alpha[b] = \{(C \to D)\}$, $\alpha[c] = \{((B \to D) \to A)\}$.
The language generated by G_1 is $L(G_1) = \{a^n(bc)^n \mid n \geq 1\}$. Figure 4 shows a derivation tree for *aabcbc*.

EXAMPLE 2.8
$G_2 = (\{a, b\}, \{A, B, C\}, A, \alpha)$,
$\alpha[a] = \{B, ((B \to A) \to (B \to C))\}$, $\alpha[b] = \{((B \to C) \leftarrow B), (C \to A)\}$.
The language generated by G_2 is $L(G_2) = \{(ab)^n \mid n \geq 2\}$. Figure 5 shows a witness derivation Δ_1 for *ababab*. We will return to this example when demonstrating how an **L**-automaton "simulates" **L**-grammar derivations.

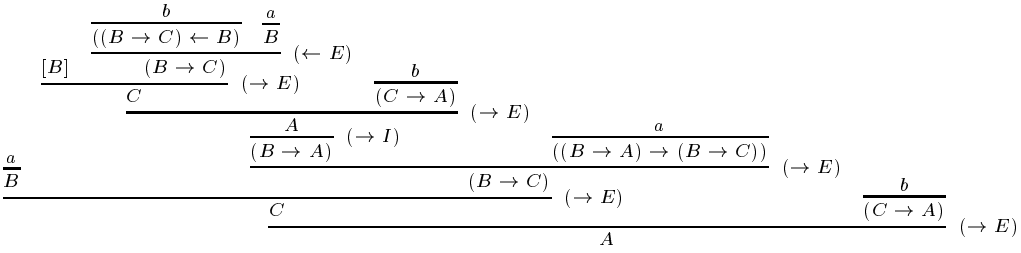


FIG. 5. A witness derivation Δ_1 for *ababab*

2.3 Assumption labeling and linearization

Since the focus is on witness derivations of **L**-grammars, rather then on arbitrary **L**-derivations, we enhance somewhat the notation of derivation trees.

Let $G = (\Sigma, \mathcal{B}, \tau_0, \alpha)$ be an **L**-grammar and let Δ be a witness derivation in G. If some node in Δ is labeled by an axiom $\tau \rhd \tau$ (without square brackets) then τ originates from the lexicon. We mark such nodes by adding an extra level to the tree. A node labeled by an axiom $\tau \rhd \tau$ has one descendant, labeled by some $\sigma \in \Sigma$ such that $\tau \in \alpha[\sigma]$. Consequently, in every sequent $\Gamma \rhd \tau$ appearing in a derivation tree Δ marked this way, Γ is of the form $[\tau_1] \dots [\tau_i]\tau_{i+1} \dots \tau_j[\tau_{j+1}] \dots [\tau_n]$, thus denoting that

- $\tau_1, \dots, \tau_i$ are *left assumptions*, discharged somewhere in Δ using $(\rightarrow I)$.
- $\tau_{i+1}, \dots, \tau_j$ are categories originating from the lexicon, i.e. there are $\sigma_{i+1}, \dots, \sigma_j \in \Sigma$, appearing at the extra level of Δ, such that $\tau_{i+1} \dots \tau_j \in \alpha[\sigma_{i+1} \dots \sigma_j]$;
- $\tau_{j+1}, \dots, \tau_n$ are *right assumptions*, discharged somewhere in Δ using $(\leftarrow I)$.

To further simplify the notation, we extend square bracketing to sequences of assumptions, thus denoting a sequence $[\tau_1] \dots [\tau_k]$ by $[\tau_1 \dots \tau_k]$. Thus, $[\Gamma]$ denotes a sequence of assumptions all of which are bracketed. Furthermore, $\hat{\Gamma}$ denotes a sequence of assumptions none of which is bracketed, namely, the assumptions originating from the lexicon.

To sum up, every sequent $\Gamma \rhd \tau$ appearing in a node of a witness derivation tree Δ of some **L**-grammar G is of the form $[\Gamma^l]\hat{\Gamma}'[\Gamma^r] \rhd \tau$, where $\Gamma' \in \alpha[w]$ (for some string $w \in \Sigma^+$ appearing at the extra level of Δ), and Γ^l, Γ^r are sequences of left and right assumptions respectively, all of which are discharged somewhere in Δ. While the whole Γ is never empty, each one of its three subsequences $\Gamma^l, \hat{\Gamma}', \Gamma^r$ may be empty.

EXAMPLE 2.9
Let $G = (\{a, b, c\}, \{A, B, C, D\}, D, \alpha)$, where $\alpha[a] = \{(C \leftarrow B)\}$, $\alpha[b] = \{(B \leftarrow A)\}$

and $\alpha[c] = \{((C \leftarrow A) \rightarrow D)\}$. Consider the following witness derivation Δ in G:

$$\dfrac{\dfrac{\dfrac{a}{(C \leftarrow B)} \quad \dfrac{\dfrac{b}{(B \leftarrow A)} \quad [A]}{B}\,(\leftarrow E)}{C}\,(\leftarrow E)}{(C \leftarrow A)}\,(\leftarrow I) \qquad \dfrac{c}{((C \leftarrow A) \rightarrow D)}}{D}\,(\rightarrow E)$$

The front (without the extra-level) of Δ is $(C \leftarrow B)\ (B \leftarrow A)\ [A]\ ((C \leftarrow A) \rightarrow D)$, and thus the bracketed assumption $[A]$ is not peripheral. However, a sequent appearing at the root of Δ is $(C \leftarrow B)\ (B \leftarrow A)\ ((C \leftarrow A) \rightarrow D) \rhd D$, i.e., $[A]$ is discharged at the moment of deriving D and thus it does not appear in the above sequent. Note also, that $[A]$ is peripheral in all the sequents where it appears, namely, $(B \leftarrow A)\ [A] \rhd B$ and $(C \leftarrow B)\ (B \leftarrow A)\ [A] \rhd C$.

Subtrees of witness derivations have similar properties. We refer to derivation-trees under the above notational conventions as *marked* derivation trees. The word appearing at the extra level of a marked tree Δ is *yield*(Δ). If $\mathrm{yield}(\Delta) = w$ and $\mathrm{root}(\Delta) = [\Gamma^l]\hat{\Gamma}'[\Gamma^r] \rhd \tau$, then $\Gamma' \in \alpha[w]$.

Note, that in all the examples until now every derivation tree has only subtrees with a bounded number of undischarged assumptions (more exactly, at most one undischarged assumption). In other words, every sequent $[\Gamma^l]\hat{\Gamma}[\Gamma^r] \rhd \tau$ appearing in the derivation trees has Γ^l, Γ^r of length≤ 1. Generally, there is no bound on the number of undischarged assumptions in every subtree. The following example demonstrates a grammar with an unbounded number of undischarged assumptions in witness derivations.

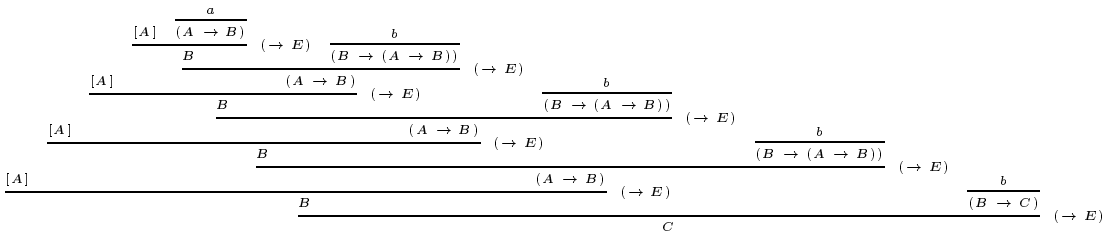
EXAMPLE 2.10
$G_3 = (\{a, b, c\}, \{A, B, C, D\}, C, \alpha)$,
where $\alpha[a] = \{(A \rightarrow B)\}$, $\alpha[b] = \{(B \rightarrow (A \rightarrow B)), (B \rightarrow C)\}$, $\alpha[c] = \{((A \rightarrow C) \rightarrow D), ((A \rightarrow D) \rightarrow C)\}$.

It is easy to see that $\{ab^{2n}c^{2n} \mid n > 0\} \subseteq L(G_3)$.
Below is a witness derivation for *abbcc*:

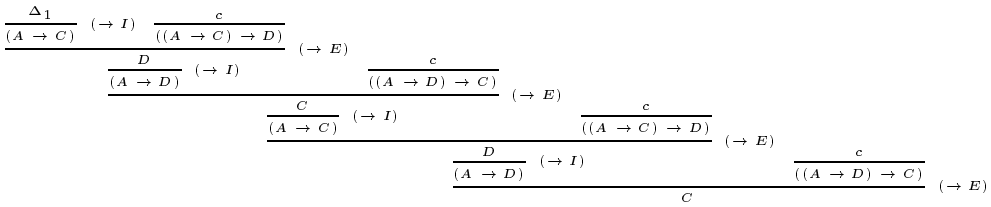
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{a}{(A \rightarrow B)}}{B}\,(\rightarrow E) \quad \dfrac{b}{(B \rightarrow (A \rightarrow B))}}{(A \rightarrow B)}\,(\rightarrow E)}{B}\,(\rightarrow E) \quad \dfrac{b}{(B \rightarrow C)}}{C}\,(\rightarrow E)}{(A \rightarrow C)}\,(\rightarrow I) \qquad \dfrac{c}{((A \rightarrow C) \rightarrow D)}}{D}\,(\rightarrow E)}{(A \rightarrow D)}\,(\rightarrow I) \qquad \dfrac{c}{((A \rightarrow D) \rightarrow C)}}{C}\,(\rightarrow E)$$

Following is a witness derivation Δ_2 for *abbbbcccc*. Its subtree Δ_1 is specified first, and then Δ_2 is specified using Δ_1.

Δ_1:



Δ_2:



Note that a subtree Δ_1 has four undischarged assumptions. For every $n \geq 1$, $ab^{2n}c^{2n}$ is accepted similarly: its witness derivation has a subtree with $2n$ undischarged assumptions; those assumptions are further discharged in the witness derivation.

The above property of **L**-grammars has an impact on a similar property of **L**-automata, which will be discussed later.

To facilitate the comparison of grammars and automata, we use a linearized notation of marked derivation trees. A linearization of a tree is the sequence of node annotations (not including the extra level), where the node annotation is the tuple *(node, node's children, node's label)*. This sequence is obtained by recursively traversing a derivation tree in preorder, starting from its leftmost leaf, and for each node advancing from its left child to its right child and then to the node itself. For convenience, terminals occurring at the extra level of the tree appear in the node annotations of their parent nodes. Thus, we obtain an order where for every application of an inference rule nodes labeled by premise(s) of the inference rule appear before a node labeled by a conclusion of the rule.

DEFINITION 2.11
(**Tree linearization**)
Let Δ be a marked **L**-derivation, having nodes $v_1, \ldots, v_n$. Then, the **linearized notation** of Δ is the sequence

$$\mathbf{lin}(\Delta) = \langle (v_1, \text{children}(v_1), \Gamma_1 \rhd \tau_1), \ldots, (v_n, \text{children}(v_n), \Gamma_n \rhd \tau_n) \rangle$$

defined recursively as follows:

- If Δ is a single leaf labeled by $[\tau] \rhd \tau$ then

  $$\text{lin}(\Delta) = \langle (v_1, \epsilon, [\tau] \rhd \tau) \rangle$$

  where ϵ denotes by abuse of notation the empty sequence of nodes.
  If Δ is labeled by the axiom $\tau \rhd \tau$ having one descendant labeled by σ (where $\tau \in \alpha[\sigma]$) then

  $$\text{lin}(\Delta) = \langle (v_1, \sigma, \tau \rhd \tau) \rangle$$

- If the root of Δ is the node v_n labeled by $\Gamma_n \rhd \tau_n$, and Δ has one subtree Δ' with the root v_k then
$$\mathrm{lin}(\Delta) = \mathrm{lin}(\Delta') \cdot \langle (v_n, (v_k), \Gamma_n \rhd \tau_n) \rangle$$
- If the root of Δ is the node v_n labeled by $\Gamma_n \rhd \tau_n$, and Δ has two subtrees Δ', Δ'' in the left-to-right order, where the node v_k is the root of Δ' and the node v_l is the root of Δ'', then
$$\mathrm{lin}(\Delta) = \mathrm{lin}(\Delta') \cdot \mathrm{lin}(\Delta'') \cdot \langle (v_n, (v_k, v_l), \Gamma_n \rhd \tau_n) \rangle$$

where '$\cdot$' denotes the concatenation of sequences of node annotations.

Following is a linearized notation of the tree Δ from example 2.9.

$$\begin{aligned}
\mathrm{lin}(\Delta) = \quad & \langle (v_1, a, (C \leftarrow B) \rhd (C \leftarrow B)), \\
& (v_2, b, (B \leftarrow A) \rhd (B \leftarrow A)), \\
& (v_3, \epsilon, [A] \rhd A), \\
& (v_4, (v_2, v_3), (B \leftarrow A)\ [A] \rhd B), \\
& (v_5, (v_1, v_4),), (C \leftarrow B)\ (B \leftarrow A)\ [A] \rhd C \\
& (v_6, (v_5), (C \leftarrow B)\ (B \leftarrow A) \rhd (C \leftarrow A)) \\
& (v_7, c, ((C \leftarrow A) \rightarrow D) \rhd ((C \leftarrow A) \rightarrow D)) \\
& (v_8, (v_6, v_7), (C \leftarrow B)\ (B \leftarrow A)\ ((C \leftarrow A) \rightarrow D) \rhd D) \rangle
\end{aligned}$$

Since in this paper every marked derivation tree Δ will be considered in its linearized notation, Δ will henceforth abbreviate $\mathrm{lin}(\Delta)$.

3 The Lambek automaton

3.1 Informal explanation

A Lambek automaton (**L**-automaton) has a storage structure, called a *hyper-stack (HS)*. A hyper-stack is a stack of *elements* each of which contains an unbounded array of *items*, of which only the two leftmost items and the rightmost item are accessible. An illustration of an **L**-automaton is given in Figure 6.

The alphabet of HS is defined over a finite set $\mathcal{O}$ of *objects*. Each element of HS contains a finite number of *items*, either objects from $\mathcal{O}$ or a *separator* '$\star$'. Every element is divided into two parts: the first part is a single object X out of $\mathcal{O}$, called the *main object*; the second part is a finite sequence β over $\mathcal{O} \cup \{\star\}$ of *hypotheses*; β itself is called the *hypotheses list (h-list)*. Note, that an h-list may contain hypotheses and '$\star$' separators.

The read head moves from left to right only; a move of the automaton is determined by at most two topmost HS elements and (possibly) the current input symbol. The move may alter the content of HS at its topmost end. Although an h-list of an HS element is unbounded, only one peripheral item from the list (plus the main object) are accessible to the automaton.

An **L**-automaton is non-deterministic. In each step the automaton performs one of the following sequences of operations:

- reads an input letter (optionally) and pushes a new element onto HS;

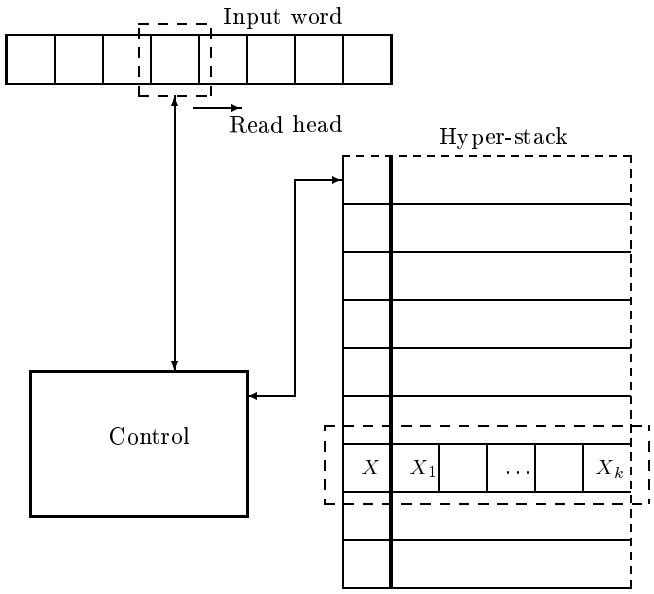


FIG. 6. The Lambek automaton

- updates the topmost element of HS by changing its main object and one peripheral item from the h-list;
- removes two topmost elements of HS and pushes a new element, whose hypotheses list is obtained by a concatenation of the lists of removed elements.

Automaton transitions are of five types:

shift This is the only transition that reads an input letter (the rest are ϵ-transitions). In a $shift$ transition a new element $(X, \star)$ (for some $X \in \mathcal{O}$) is pushed onto HS. The h-list of the new element contains only a separator '$\star$'.

hypothesis-shift This is an ϵ-transition that pushes an element (X, X) (for some $X \in \mathcal{O}$) onto HS. The h-list of the new element contains a single hypothesis X.

reduce This is an ϵ-transition that pops two topmost elements (X, β_1) and (Y, β_2) from HS and pushes a new element (Z, γ) (for some $Z \in \mathcal{O}$). The h-list γ of the new element is obtained by concatenation of lists β_1 and β_2 under the following convention, avoiding occurrence of successive separator signs ('$\star$') in β: if $\beta_1 = \gamma_1\star$ and $\beta_2 = \star\gamma_2$, then $\gamma = \gamma_1 \star \gamma_2$; otherwise, $\gamma = \beta_1\beta_2$.

left-release This is an ϵ-transition that updates the topmost element (X, β) in HS to be (X', β') (for some $X' \in \mathcal{O}$), where β' is obtained by removing the leftmost hypothesis [4] from β.

right-release Similarly, but with the rightmost hypothesis.

Different types of transitions are illustrated in Figure 7.

The automaton starts its run with HS empty and with its read head at the beginning of the input. It takes a sequence of moves (or steps). Each step depends on the

[4] If the leftmost value of β is '$\star$', the $left-release$ transition is undefined.

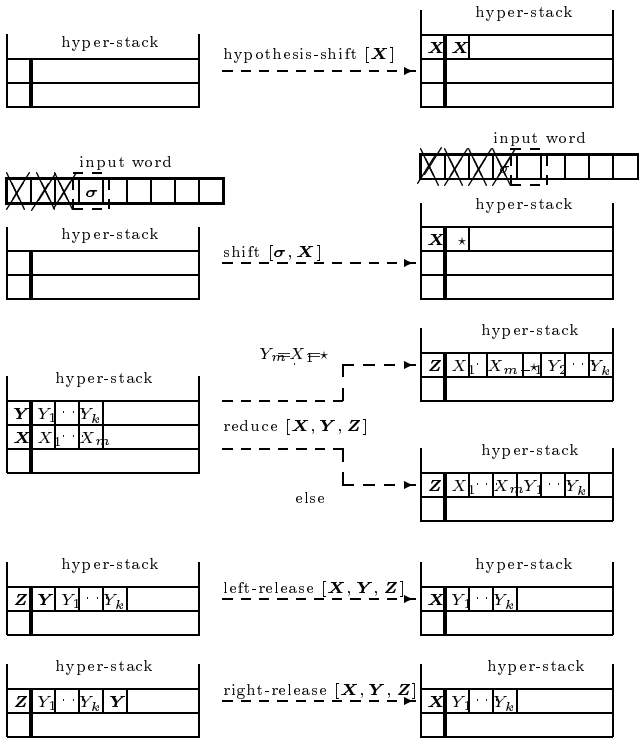


FIG. 7. Transition types of the **L**-automaton

topmost symbols in HS and possibly on the next input letter. At each step it performs one of the five transition types, described above. If a *reduce* transition is defined for the two topmost elements in HS, then the automaton may perform it, thus decreasing the number of elements in HS. However, the automaton may alternatively perform a $shift/hypothesis-shift$ transition, thus increasing the number of elements in HS, or it may take $left/right-release$ transition (if the latter are defined), thus decreasing the number of hypotheses in the h-list of the topmost element. If more then one transition is defined at a certain point, then the automaton makes a nondeterministic choice between possible transitions.

The automaton accepts the input string when it is exhausted and HS contains a single element $(X_0, \star)$ (with an h-list containing only a separator '$\star$'), where X_0 is the distinguished object in $\mathcal{O}$. Note that not every run terminates. For example, the automaton may systematically choose to perform hypothesis-shift steps (which do not depend on the topmost element(s) of HS), thus running into an infinite sequence of moves, without even reading some of its input.

3.2 Formalizing the automaton

A formal definition of an **L**-automaton is given below. Automaton transitions are defined in two steps: first, all the possible *transition types* are specified; next, several

constraints are imposed on the transitions, thus reducing a transition set to a subset of *legal transitions*. The constraints on the automaton transitions simulate the constraints on the derivation relation of **L**-grammars, assimilating arbitrary objects to categories; this will be explained in detail in section 4.

Transitions depend on arguments (in square brackets). Some transitions have variables that do not appear in their argument list (for example, β, γ in the *reduce* transition). Those are free variables whose values range over the whole set of possible values (in other words, values of those variables are not relevant for specifying the transition). In general, every transition of an **L**-automaton considers at most two *peripheral hypotheses* in the h-list; the rest of hypotheses (for example, β, γ in the *reduce* transitions) are simply copied to the h-list of the uppermost element in HS.

Thus, in order to define a transition relation δ of a *specific* **L**-automaton, it suffices to supply arguments for all the automaton's transitions, while ensuring that the transition constraints are satisfied. For example, in order to define all the $shift$ transitions of an **L**-automaton, it suffices to supply all the pairs of arguments $[\sigma \in \Sigma, X \in \mathcal{O}]$ required for definition of the $shift$ transition below.

DEFINITION 3.1
A **L**-automaton is a tuple $M = (\Sigma, \mathcal{O}, X_0, \delta)$, where:

- Σ is a finite input alphabet.
- $\mathcal{O}$ is a finite set of objects, ranged over by $X, X_i, Y, Y_i, Z, Z_i, \ldots$ (where $\star \notin \mathcal{O}$).
- $X_0 \in \mathcal{O}$ is the distinguished object.
- δ is the transition function, defined as follows:

$$\delta : (\{\epsilon\} \cup (\mathcal{O} \cup \{\star\})^2 \cup (\mathcal{O} \cup \{\star\})^4) \times (\Sigma \cup \{\epsilon\}) \longrightarrow \mathcal{P}_{fin}(\bigcup_{i=1\ldots3} (\mathcal{O} \cup \{\star\})^i)$$

Note, that the two or four items in the domain of δ in the above definition are not successive items in HS. Actually, the two items are a main object and one peripheral item (either a hypothesis or '$\star$') in the topmost element of HS; likewise, the four items are two pairs of main object + peripheral item in the two topmost elements of HS.

We use the following notation in transition definitions:
– $(X, Y) \in \delta(\epsilon, \sigma)$ stands for the transition that pushes an element (X, Y) onto HS on an input letter σ;
– $(X, \beta X_r) \in \delta((Y, X_l \beta X_r), \epsilon)$ stands for an ϵ-transition that updates the topmost element in HS by removing its leftmost hypothesis X_l from the h-list $X_l \beta X_r$ and changing the main object from Y to X.
– $(Z, \beta N_r \gamma) \in \delta((X, \beta N_r)(Y, N_l \gamma), \epsilon)$ stands for an ϵ-transition that pops the two topmost elements from HS and replaces them with a new element, whose main object is Z and whose h-list is the concatenation of h-lists of the removed elements, not including the leftmost item N_l of the topmost removed element (here, N_l, N_r range over $\mathcal{O} \cup \{\star\}$).

In this notation the hyper-stack grows from left to right.

The automaton has several types[5] of transitions, described below, on which con-

[5] An actual transition is defined by supplying values to arguments (in square brackets) of one of the transition types.

straints will be imposed.

shift $[\sigma \in \Sigma, X \in \mathcal{O}]$:	$(X, \star) \in \delta(\epsilon, \sigma)$	
hypothesis-shift $[X \in \mathcal{O}]$:	$(X, X) \in \delta(\epsilon, \epsilon)$	
reduce $[X, Y, Z \in \mathcal{O}]$:	$(Z, \beta \star \gamma) \in \delta((X, \beta\star)(Y, \star\gamma), \epsilon)$ $(Z, \beta N_r N_l \gamma) \in \delta((X, \beta N_r)(Y, N_l\gamma), \epsilon)$	 where $N_r \neq \star$ or $N_l \neq \star$
left-release $[X, Y, Z \in \mathcal{O}]$:	$(X, \beta) \in \delta((Z, Y\beta), \epsilon)$	where $\beta \neq \epsilon$
right-release $[X, Y, Z \in \mathcal{O}]$:	$(X, \beta) \in \delta((Z, \beta Y), \epsilon)$	where $\beta \neq \epsilon$

In order to formulate the constraints on automaton transitions, we define the *equivalence relation* $R_\approx \subseteq \mathcal{O} \times \mathcal{O}$ induced by M. $R_\approx$ is the *smallest* set satisfying the following conditions:

- For all $X \in \mathcal{O}$: $(X, X) \in R_\approx$
- For all $X_1, X_2, Y_1, Y_2, Z_1, Z_2 \in \mathcal{O}$ such that left-release$[X_1, Y_1, Z_1]$, left-release$[X_2, Y_2, Z_2]$ or right-release$[X_1, Y_1, Z_1]$, right-release$[X_2, Y_2, Z_2]$:

  $$(X_1, X_2) \in R_\approx \text{ implies } (Y_1, Y_2) \in R_\approx \text{ and } (Z_1, Z_2) \in R_\approx$$

  $$(Y_1, Y_2) \in R_\approx \text{ and } (Z_1, Z_2) \in R_\approx \text{ implies } (X_1, X_2) \in R_\approx$$

- For all $X, Y, Z \in \mathcal{O}$: $(X, Y) \in R_\approx$ and $(Y, Z) \in R_\approx$ implies $(X, Z) \in R_\approx$

Let Π be the set of all the equivalence classes of $R_\approx$: $\Pi = \bigcup_{X \in \mathcal{O}} [X]$, and let I_Π be the identity relation over Π: $I_\Pi = \{([X], [X]) \mid [X] \in \Pi$.

Next, we recursively define a *dependency relation* of M, $D_M \subseteq \Pi \times \Pi$. D_M is the least set satisfying the following conditions:

- For all $X, Y, Z \in \mathcal{O}$:
  left-release$[X, Y, Z]$ or right-release$[X, Y, Z]$ imply $([X], [Y]), ([X], [Z]) \in D_M$
- For all $X, Y, Z \in \mathcal{O}$:
  $([X], [Y]) \in D_M$ and $([Y], [Z]) \in D_M$ imply $([X], [Z]) \in D_M$

Note that in contrast to $R_\approx$, D_M is not reflexive and not symmetric.

Finally, we define constraints on automaton transitions; these constraints must be satisfied by every **L**-automaton:

1. For every $[X] \in \Pi$ there is $X_1 \in [X]$ such that hypothesis-shift$[X_1]$
2. For all $X_1, Y_1, Z_1, X_2, Y_2, Z_2 \in \mathcal{O}$:
   left-release$[X_1, Y_1, Z_1]$ and right-release$[X_2, Y_2, Z_2]$ imply $[X_1] \neq [X_2]$
3. For all $X_1, Y_1, Z_1 \in \mathcal{O}, X_2 \in [X_1], Y_2 \in [Y_1], Z_2 \in [Z_1]$:
   left-release$[X_1, Y_1, Z_1]$ iff left-release$[X_2, Y_2, Z_2]$, and
   right-release$[X_1, Y_1, Z_1]$ iff right-release$[X_2, Y_2, Z_2]$

4. For all $X, Y, Z \in \mathcal{O}$:
   reduce$[X, Y, Z]$ iff either left-release$[Y, X, Z]$ or right-release$[X, Y, Z]$
5. $D_M \cap I_\Pi = \emptyset$

The nature and the purpose of automaton constraints will be discussed in section 4.2.

Some transition types (namely, *reduce*, $left - release$ and $right - release$) have side conditions. Thus, *release* transitions require that the h-list contains at least one more item (either an object from $\mathcal{O}$ or $\star$) except the released hypothesis; this implies that the last hypothesis in the list cannot be released, unless an input terminal was read.

As usual for an automaton, a configuration contains all components relevant to the future steps of the automaton. For an **L**-automaton, this means the HS contents and the remaining input.

DEFINITION 3.2
Let $\Phi = (\mathcal{O} \cup \{\star\})^\star$, ranged over by $\beta, \beta_i, \gamma, \gamma_i$.
Let Θ be the set of all the possible elements of HS: $\Theta = \mathcal{O} \times \Phi$, ranged over[6] by μ, μ_i, ν, ν_i.

A **configuration** of an **L**-automaton M is the pair $\langle w, \mu \rangle \in \Sigma^\star \times \Theta^+$, where w is the remaining input and μ is the content of HS.

Moves are represented by the binary relation $\vdash$ between configurations, which is defined as follows:
$\langle \boldsymbol{aw, \mu} \rangle \vdash \langle \boldsymbol{w, \mu'} \rangle$ iff $\mu = \mu_1\mu_2$, $\mu' = \mu_1\mu_3$ and $\mu_3 \in \delta(\mu_2, a)$ for $\mu_1, \mu_2 \in \Theta^\star$, $\mu_3 \in \Theta$, $a \in \Sigma \cup \{\epsilon\}$.

For configurations c and c' we define $c \vdash^n c'$ if there exist configurations $c_1, \ldots, c_{n+1}$ such that $c_1 = c$, $c_{n+1} = c'$ and $c_i \vdash c_{i+1}$ for $1 \leq i \leq n$.

$\vdash^\star$, the reflexive transitive closure of $\vdash$, then stands for $\bigcup_{n \geq 0} \vdash^n$, and $\vdash^+$ stands for $\bigcup_{n \geq 1} \vdash^n$.

For an arbitrary $w \in \Sigma^\star$, $c_{ini} = \langle w, \epsilon \rangle$ is called an **initial configuration** (by abuse of notation, we denote an empty sequence over Θ by ϵ).

The configuration $c_{acc} = \langle \epsilon, (X_0, \star) \rangle$, where X_0 is the distinguished object, is called the **accepting configuration**.

A word $w \in \Sigma^\star$ is **accepted** by M, if $\langle w, \epsilon \rangle \vdash^\star c_{acc}$.

The **language** of M, is $L(M) = \{w \in \Sigma^\star \mid w \text{ is accepted by } M\}$.

A **run** π of the automaton is a sequence of moves, where every two successive moves are in the relation $\vdash$:

$$\pi = c_0 \vdash c_1 \vdash c_2 \vdash \ldots$$

Note, that being nondeterministic, M may have more than one run on a given input word, some of which are infinite. The example below shows an **L**-automaton accepting the language $\{(ab)^n \mid n \geq 2\}$.

EXAMPLE 3.3

$$M_1 = (\{a, b\}, \{X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8\}, X_0, \delta)$$

where δ includes the following transitions:

- shift transitions:

[6] Sometimes μ, μ_i, ν, ν_i will denote *sequences* of elements over Θ.

– shift$[a, X_8]$: $(X_8, \star) \in \delta(\epsilon, a)$
– shift$[b, X_2]$: $(X_2, \star) \in \delta(\epsilon, b)$
– shift$[b, X_5]$: $(X_5, \star) \in \delta(\epsilon, b)$
– shift$[a, X_7]$: $(X_7, \star) \in \delta(\epsilon, a)$

- hypothesis-shift transitions: $\forall\ 0 \leq i \leq 7$:
  hypothesis-shift$[X_i]$: $(X_i, X_i) \in \delta(\epsilon, \epsilon)$
- reduce transitions: for $X \neq \star$ or $Y \neq \star$:
  – reduce$[X_2, X_1, X_3]$: $(X_3, \beta \star \gamma) \in \delta((X_2, \beta\star)(X_1, \star\gamma), \epsilon)$

  $(X_3, \beta XY\gamma) \in \delta((X_2, \beta X)(X_1, Y\gamma), \epsilon)$

  – reduce$[X_2, X_8, X_3]$: $(X_3, \beta \star \gamma) \in \delta((X_2, \beta\star)(X_8, \star\gamma), \epsilon)$

  $(X_3, \beta XY\gamma) \in \delta((X_2, \beta X)(X_8, Y\gamma), \epsilon)$

  – reduce$[X_1, X_3, X_4]$: $(X_4, \beta \star \gamma) \in \delta((X_1, \beta\star)(X_3, \star\gamma), \epsilon)$

  $(X_4, \beta XY\gamma) \in \delta((X_1, \beta X)(X_3, Y\gamma), \epsilon)$

  – reduce$[X_8, X_3, X_4]$: $(X_4, \beta \star \gamma) \in \delta((X_8, \beta\star)(X_3, \star\gamma), \epsilon)$

  $(X_4, \beta XY\gamma) \in \delta((X_8, \beta X)(X_3, Y\gamma), \epsilon)$

  – reduce$[X_4, X_5, X_0]$: $(X_0, \beta \star \gamma) \in \delta((X_4, \beta\star)(X_5, \star\gamma), \epsilon)$

  $(X_0, \beta XY\gamma) \in \delta((X_4, \beta X)(X_5, Y\gamma), \epsilon)$

  – reduce$[X_6, X_7, X_3]$: $(X_3, \beta \star \gamma) \in \delta((X_6, \beta\star)(X_7, \star\gamma), \epsilon)$

  $(X_3, \beta XY\gamma) \in \delta((X_6, \beta X)(X_7, Y\gamma), \epsilon)$

  – reduce$[X_1, X_6, X_0]$: $(X_0, \beta \star \gamma) \in \delta((X_1, \beta\star)(X_6, \star\gamma), \epsilon)$

  $(X_0, \beta XY\gamma) \in \delta((X_1, \beta X)(X_6, Y\gamma), \epsilon)$

  – reduce$[X_8, X_6, X_0]$: $(X_0, \beta \star \gamma) \in \delta((X_8, \beta\star)(X_6, \star\gamma), \epsilon)$

  $(X_0, \beta XY\gamma) \in \delta((X_8, \beta X)(X_6, Y\gamma), \epsilon)$

- left-release transitions: for $\beta \neq \epsilon$:
  – left-release$[X_3, X_1, X_4]$: $(X_3, \beta) \in \delta((X_4, X_1\beta), \epsilon)$
  – left-release$[X_3, X_8, X_4]$: $(X_3, \beta) \in \delta((X_4, X_8\beta), \epsilon)$
  – left-release$[X_5, X_4, X_0]$: $(X_5, \beta) \in \delta((X_0, X_4\beta), \epsilon)$
  – left-release$[X_6, X_1, X_0]$: $(X_6, \beta) \in \delta((X_0, X_1\beta), \epsilon)$
  – left-release$[X_6, X_8, X_0]$: $(X_6, \beta) \in \delta((X_0, X_8\beta), \epsilon)$
  – left-release$[X_7, X_6, X_3]$: $(X_7, \beta) \in \delta((X_3, X_6\beta), \epsilon)$
- right-release transitions: for $\beta \neq \epsilon$:
  – right-release$[X_2, X_1, X_3]$: $(X_2, \beta) \in \delta(X_3, \beta X_1), \epsilon)$
  – right-release$[X_2, X_8, X_3]$: $(X_2, \beta) \in \delta(X_3, \beta X_8), \epsilon)$

First, we verify that M_1 is a **L**-automaton, i.e., that the constraints introduced in definition 3.1 are satisfied. The equivalence relation $R_{\approx}$ is:

$$R_{\approx} = \{(X_0,X_0),(X_1,X_1),(X_2,X_2),(X_3,X_3),(X_4,X_4),(X_5,X_5),(X_6,X_6),(X_7,X_7),\\ (X_8,X_8),(X_1,X_8),(X_8,X_1)\}$$

Note the non-trivial pair (X_1, X_8) in $R_{\approx}$.

$$\Pi = \{[X_0],[X_1],[X_2],[X_3],[X_4],[X_5],[X_6],[X_7]\}$$

Verification of the five constraints on the automaton transitions:

1. For every $0 \leq i \leq 7$, hypothesis-shift$[X_i]$;
   as to $[X_8]$, $X_1 \in [X_8]$
2. The first argument in the left-release transitions in δ is either X_3, X_5, X_6 or X_7;
   the first argument in the right-release transitions in δ is X_2;
   $[X_2] \neq [X_3]$, $[X_2] \neq [X_5]$, $[X_2] \neq [X_6]$, $[X_2] \neq [X_7]$
3. The only non-trivial equivalence class is $[X_1] = [X_8]$.
   left-release$[X_3, X_1, X_4]$ and left-release$[X_3, X_8, X_4]$;
   left-release$[X_6, X_1, X_0]$ and left-release$[X_6, X_8, X_0]$;
   finally, right-release$[X_2, X_1, X_3]$ and right-release$[X_2, X_8, X_3]$
4. check all the reduce transitions in δ:
   reduce$[X_2, X_1, X_3]$ and right-release$[X_2, X_1, X_3]$;
   reduce$[X_2, X_8, X_3]$ and right-release$[X_2, X_8, X_3]$;
   reduce$[X_1, X_3, X_4]$ and left-release$[X_3, X_1, X_4]$;
   reduce$[X_8, X_3, X_4]$ and left-release$[X_3, X_8, X_4]$;
   reduce$[X_4, X_5, X_0]$ and left-release$[X_5, X_4, X_0]$;
   reduce$[X_6, X_7, X_3]$ and left-release$[X_7, X_6, X_3]$;
   reduce$[X_1, X_6, X_0]$ and left-release$[X_6, X_1, X_0]$;
   reduce$[X_8, X_6, X_0]$ and left-release$[X_6, X_8, X_0]$
5. $$D_M = \{([X_2],[X_1]),([X_2],[X_3]),([X_3],[X_1]),([X_3],[X_4]),([X_5],[X_4]),([X_5],[X_0]),\\ ([X_7],[X_6]),([X_7],[X_3]),([X_6],[X_1]),([X_6],[X_0]),([X_2],[X_4]),([X_7],[X_1]),\\ ([X_7],[X_4])\}$$
   $$I_\Pi = \{([X_0],[X_0]),([X_1],[X_8]),([X_2],[X_2]),([X_3],[X_3]),([X_4],[X_4]),([X_5],[X_5]),\\ ([X_6],[X_6]),([X_7],[X_7])\}$$
   $$D_M \cap I_\Pi = \emptyset$$

Figure 8 shows an accepting run π_1 of M_1 for the word '*ababab*'.

3.3 Properties of the **L***-automaton*

We explore here some properties of **L**-automata, which will be used in the next sections. First, we show that the maximal number of separators ('$\star$') in an HS element never decreases during a run. For the next proposition we use the following notation.

DEFINITION 3.4
For $\beta \in (\mathcal{O} \cup \{\star\})^\star$, $\#_\star(\beta)$ denotes the number of '$\star$'s in β.

$$
\begin{array}{lll}
\pi_1 = & \langle ababab, \epsilon\rangle \vdash & \text{shift } [a, X_8] \\
& \langle babab, (X_8, \star)\rangle \vdash & \text{hypothesis-shift } [X_1] \\
& \langle babab, (X_8, \star)(X_1, X_1)\rangle \vdash & \text{shift } [b, X_2] \\
& \langle abab, (X_8, \star)(X_1, X_1)(X_2, \star)\rangle \vdash & \text{shift } [a, X_8] \\
& \langle bab, (X_8, \star)(X_1, X_1)(X_2, \star)(X_8, \star)\rangle \vdash & \text{reduce } [X_2, X_8, X_3] \\
& \langle bab, (X_8, \star)(X_1, X_1)(X_3, \star)\rangle \vdash & \text{reduce } [X_1, X_3, X_4] \\
& \langle bab, (X_8, \star)(X_4, X_1\star)\rangle \vdash & \text{shift } [b, X_5] \\
& \langle ab, (X_8, \star)(X_4, X_1\star)(X_5, \star)\rangle \vdash & \text{reduce } [X_4, X_5, X_0] \\
& \langle ab, (X_8, \star)(X_0, X_1\star)\rangle \vdash & \text{left-release } [X_6, X_1, X_0] \\
& \langle ab, (X_8, \star)(X_6, \star)\rangle \vdash & \text{shift } [a, X_7] \\
& \langle b, (X_8, \star)(X_6, \star)(X_7, \star)\rangle \vdash & \text{reduce } [X_6, X_7, X_3] \\
& \langle b, (X_8, \star)(X_3, \star)\rangle \vdash & \text{reduce } [X_8, X_3, X_4] \\
& \langle b, (X_4, \star)\rangle \vdash & \text{shift } [b, X_5] \\
& \langle \epsilon, (X_4, \star)(X_5, \star)\rangle \vdash & \text{reduce } [X_4, X_5, X_0] \\
& \langle \epsilon, (X_0, \star)\rangle &
\end{array}
$$

FIG. 8. An accepting run of M_1 for the string '*ababab*'

PROPOSITION 3.5
(**M-invariant**)
For every **L**-automaton move $\langle v, (X_1, \beta_1) \dots (X_k, \beta_k))\rangle \vdash \langle w, (B_1, \gamma_1) \dots (B_m, \gamma_m)\rangle$,

$$MAX_{i=1,\dots,k}\ \#_\star(\beta_i) \leq MAX_{j=1,\dots,m}\ \#_\star(\gamma_j)$$

PROOF. The proof is by case analysis over transition types.

- $shift$: $\langle v, (X_1, \beta_1) \dots (X_k, \beta_k))\rangle \vdash \langle w, (X_1, \beta_1) \dots (X_k, \beta_k)(X, \beta)\rangle$.
  The element (X, β) is added into HS, thus satisfying the claim.
- $hypothesis - shift$: similar.
- $reduce$:
  $\langle v, (X_1, \beta_1) \dots (X_{k-2}, \beta_{k-2})(X_{k-1}, \beta_{k-1})(X_k, \beta_k))\rangle \vdash \langle v, (X_1, \beta_1) \dots (X_{k-2}, \beta_{k-2})(X, \beta)\rangle$.
  Let $MAX_{1\leq i\leq k}\#(\beta_i) = n$. If $\#_\star(\beta_{k-1}) < n$ and $\#_\star(\beta_k) < n$, then the claim trivially holds. Suppose without loss of generality, that $\#_\star(\beta_{k-1}) = n$. There are two possibilities:
  1. If $\beta_{k-1} = \gamma_1\star$ and $\beta_k = \star\gamma_2$ then by the *reduce* transition definition $\beta = \gamma_1 \star \gamma_2$ and thus $\#_\star(\beta) \geq n$, thus satisfying the claim.
  2. Otherwise, $\beta = \beta_{k-1}\beta_k$, and thus again $\#_\star(\beta) \geq n$.
- $left{-}release$: $\langle v, (X_1, \beta_1) \dots (X_{k-1}, \beta_{k-1})(X_k, \beta_k))\rangle \vdash \langle v, (X_1, \beta_1) \dots (X_{k-1}, \beta_{k-1})(X, \beta)\rangle$.
  Let $MAX_{1\leq i\leq k}\#(\beta_i) = n$. If $\#_\star(\beta_k) < n$, then the claim trivially holds.
  Suppose that $\#_\star(\beta_k) = n$. By the $left - release$ transition definition $\beta_k = Y\beta$ where $Y \in \mathcal{O}$, thus implying that $\#_\star(\beta) = n$.
- $right - release$: similar.

∎

The above proposition implies an important property of the structure of h-lists during accepting runs.

PROPOSITION 3.6
Let $\pi = \langle u, \epsilon\rangle \vdash^n \langle v, (X_1, \beta_1)\dots(X_k, \beta_k)\rangle$ be a subrun of an accepting run. Then for every $1 \leq i \leq k$:

- $\beta_i = \gamma_1 \star \gamma_2$, where $\gamma_1, \gamma_2 \in \mathcal{O}^\star$

or

- $\beta_i \in \mathcal{O}^+$.

PROOF. The proof is by induction on n, the length of π.

$n = 1$: the first step is either $shift$ or $hypothesis - shift$ (the rest of the transitions are undefined when the automaton is in its initial configuration). Consequently, after the first step HS contains either $(X, \star)$ or (X, X), thus satisfying the conditions.

Suppose the claim holds for runs of length$\leq n$. Consider a run of length $n+1$. Let $(X_1, \beta_1)\dots(X_k, \beta_k)$ be the content of HS just before the last step of the run. By the induction hypothesis the claim is satisfied for every $1 \leq i \leq k$. There are several possibilities for the $n+1$-th step:

- $shift$: after this step HS contains $(X_1, \beta_1)\dots(X_k, \beta_k)(X, \star)$, which verifies the claim.
- $hypothesis - shift$: after this step HS contains $(X_1, \beta_1)\dots(X_k, \beta_k)(X, X)$, which verifies the claim.
- $reduce$: $(X_1, \beta_1)\dots(X_{k_2}, \beta_{k-2})(X_{k-1}, \beta_{k-1})(X_k, \beta_k) \vdash (X_1, \beta_1)\dots(X_{k_2}, \beta_{k-2})(X, \beta)$. By the induction hypothesis, $\beta_{k-1} \neq \epsilon$ and $\beta_k \neq \epsilon$, implying that $\beta \neq \epsilon$. Furthermore, since π is a subrun of an accepting run and since $c_{acc} = \langle \epsilon, \star\rangle$, then by proposition 3.5, $\#_\star(\beta) \leq \#_\star(\star) = 1$. Thus, either $\beta = \gamma_1 \star \gamma_2$ ($\gamma_1\gamma_2 \in \mathcal{O}^\star$) or $\beta \in \mathcal{O}^+$.
- $left - release$: $(X_1, \beta_1)\dots(X_{k_1}, \beta_{k-1})(X_k, \beta_k) \vdash (X_1, \beta_1)\dots(X_{k_1}, \beta_{k-1})(X, \beta)$. The left-release transition definition requires that $\beta \neq \epsilon$. Furthermore, since π is a subrun of an accepting run and since $c_{acc} = \langle \epsilon, \star\rangle$, then by proposition 3.5, $\#_\star(\beta) \leq \#_\star(\star) = 1$. Thus, either $\beta = \gamma_1 \star \gamma_2$ ($\gamma_1\gamma_2 \in \mathcal{O}^\star$) or $\beta \in \mathcal{O}^+$.
- $right - release$: similar.

■

The next two properties are inherent to every automaton having a stack. We relate them to the **L**-automaton, whose hyper-stack is a generalization of a stack.

- *wrapping* a run: adding elements at the bottom of HS and adding more input at the end of a given input does not affect a given sequence of moves; the effect of these operations is like 'wrapping' a run by a bigger run, i.e., it is like the automaton does some moves (filling HS) before a run starts, and some moves (reading more input) after a run ends.
- *unwrapping* a run: if a given sequence of moves does not change some lower elements in HS, and if it does not read the whole input, then the untouched elements may be removed from HS and the unread input may be removed, resulting in the same sequence of moves; the effect of these operations is like 'unwrapping' a run out of a bigger run, s.t. the resulting run starts at the initial configuration and reads the whole input.

PROPOSITION 3.7
(Wrapping a run)
If $\pi = \langle u, \epsilon\rangle \vdash^k \langle \epsilon, (X_1, \beta_1)\dots(X_n, \beta_n)\rangle$ is a run of M, then:

1. (adding more input after the given one):
   for every $v \in \Sigma^\star$ there is a run $\pi' = \langle uv, \epsilon\rangle \vdash^k \langle v, (X_1, \beta_1)\dots(X_n, \beta_n)\rangle$,
   where for every $0 \leq i \leq k$,
   if $c_i = \big\langle w, (X_1^i, \beta_1^i)\dots(X_n^i, \beta_n^i)\big\rangle$ then $c_i' = \big\langle wv, (X_1^i, \beta_1^i)\dots(X_n^i, \beta_n^i)\big\rangle$.
2. (adding elements at the bottom of HS):
   for every $\mu_1, \dots, \mu_m \in \Theta$ $(m \geq 1)$ there is a run
   $\pi'' = \langle u, \mu_1\dots\mu_m\rangle \vdash^k \langle \epsilon, \mu_1\dots\mu_m \cdot (X_1, \beta_1)\dots(X_n, \beta_n)\rangle$,
   where for every $0 \leq i \leq k$,
   if $c_i = \big\langle u, (X_1^i, \beta_1^i)\dots(X_n^i, \beta_n^i), \big\rangle$ then $c_i'' = \big\langle u, \mu_1\dots\mu_m \cdot (X_1^i, \beta_1^i)\dots(X_n^i, \beta_n^i)\big\rangle$.

PROPOSITION 3.8
(Unwrapping a run)
If $\pi = \langle uv, \mu_1\dots\mu_m\rangle \vdash^k \langle v, \mu_1\dots\mu_m \cdot (X_1, \beta_1)\dots(X_n, \beta_n)\rangle$ $(k \geq 0)$ is a run of M, where for every $0 \leq i \leq k$, $c_i = \big\langle u^i v, \mu_1\dots\mu_m\nu_i\big\rangle$ for some $u^i \in \Sigma^\star$, $\nu_i \in \Theta^\star$, then:

1. (removing the unread input):
   there is a run $\pi' = \langle u, \mu_1\dots\mu_m\rangle \vdash^k \langle \epsilon, \mu_1\dots\mu_m \cdot (X_1, \beta_1)\dots(X_n, \beta_n)\rangle$,
   where for every $0 \leq i \leq k$, $c_i' = \big\langle u^i, \mu_1\dots\mu_m\nu_i\big\rangle$.
2. (removing the untouched lower items from HS):
   there is a run $\pi'' = \langle uv, \epsilon\rangle \vdash^k \langle v, (X_1, \beta_1)\dots(X_n, \beta_n)\rangle$,
   where for every $0 \leq i \leq k$, $c_i'' = \big\langle u^i v, \nu_i\big\rangle$.

The proofs of both propositions are by a straightforward induction on the length of π; they are similar to the proofs of the corresponding theorems for pushdown automata and are omitted here.

Note, that not every sequence of moves may be unwrapped, but only those sequences that do not change one or more lower elements of HS or do not read the whole input. Let us focus on runs that do not touch lower elements of HS and observe some properties of such runs.

Consider an accepting run π. Vacuously, it does not change the lower elements in HS, since in the initial configuration HS is empty. Furthermore, accepting runs have a property relating to HS content: in the initial configuration c_{ini} HS is empty, whereas in the accepting configuration c_{acc} it contains the single element $(X_0, \star)$; thus, the total number of elements in HS is increased by one, though during the run HS might have contained more elements. In general, not only accepting runs have the above properties. For example, we may wrap an accepting run π by a bigger run (adding k elements at the bottom of HS and, possibly, adding more input). The resulting run has the same moves as in π; consequently, the resulting run will also increase the number of elements in HS by one; furthermore, it will not change k lower elements in HS. We call such runs *balanced* runs; namely, balanced runs are runs that do not change already existing elements in HS and increase by one a number of elements in HS.

DEFINITION 3.9
A **balanced** run of an **L**-automaton is a sequence of moves $\pi = c_i \vdash^+ c_j$ of the form $\langle w_i, \mu\rangle \vdash \langle w_{i+1}, \mu\nu_1\rangle \vdash^\star \langle w_j, \mu \cdot (X, \beta)\rangle$,

where $\mu \in \Theta^\star$, and for every $i \leq k \leq j$, $c_k = \langle w_k, \mu\nu_k \rangle$ for some $w_k \in \Sigma^\star$ and some $\nu_k \in \Theta^+$.

PROPOSITION 3.10
Every finite sequence of moves of an **L**-automaton may be partitioned into a sequence of balanced runs.

PROOF. Consider a sequence of moves $\pi = c_0 \vdash^+ c_k$. Let $c_k = \langle w_k, (X_1, \beta_1) \ldots (X_n, \beta_n) \rangle$, $n \geq 1$. Then, for every $1 \leq i \leq n$ there is the minimal j_i $(1 \leq j_i \leq k)$, such that after the j_i-th move the i lower elements $(X_1, \beta_1) \ldots (X_i, \beta_i)$ do not change. Thus, π may be partitioned into n sequences $\pi_1 \ldots \pi_n$:

$$\pi_1 = c_0 \vdash^+ c_{j_1},\ \pi_2 = c_{j_1} \vdash^+ c_{j_2},\ \ldots,\ \pi_{n-1} = c_{j_{n-2}} \vdash^+ c_{j_{n-1}},\ \pi_n = c_{j_{n-1}} \vdash^+ c_k$$

where every π_i increases by one the number of elements in HS, and the elements that are in HS at the beginning of π_i are not changed during π_i. Note, that all the runs π_i may be unwrapped (by proposition 3.7), resulting in runs starting from the initial configuration c_{ini} and reading the whole input. ∎

PROPOSITION 3.11
(Properties of balanced runs)

1. Every balanced run starts with a shift or hypothesis-shift transition.
2. A balanced run of length≥ 2 does not end with shift or hypothesis-shift transition.

PROOF. Let $\pi = \langle u, \mu_1 \ldots \mu_n \rangle \vdash^k \langle v, \mu_1 \ldots \mu_n \nu \rangle$ be a balanced run, where $\mu_i, \nu \in \Theta$ $(1 \leq i \leq n)$.

1. Since π is balanced, it does not change the initial content of HS, $\mu_1 \ldots \mu_n$. Consequently, the first step of π cannot be *reduce* (because it would remove μ_n from HS), and it cannot be $left-release/right-release$ (because they would change μ_n in HS). Thus, the only possibilities are $shift$ and $hypothesis-shift$.
2. Suppose π is of length≥ 2 and the last move of π is $shift$ or $hypothesis-shift$. Since $shift$ and $hypothesis-shift$ add ν into HS, then the rest of moves do not increase the number of elements in HS; thus, those moves end with HS containing only $\mu_1 \ldots \mu_n$. By (1), those moves begin with $shift$ or $hypothesis-shift$; furthermore, they must remove all the elements that they pushed over $\mu_1 \ldots \mu_n$. However, when removing the last element over $\mu_1 \ldots \mu_n$, *reduce* removes also μ_n, which contradicts the fact that π is a balanced run.

∎

4 Relating L-automata and L-grammars

In this section we define a notion of *tight relation* between **L**-automata and **L**-grammars, such that every **L**-automaton is tightly related to some **L**-grammar and vice versa, every **L**-grammar is tightly related to some **L**-automaton. The pairs automaton+grammar in this relation satisfy the following property: *every accepting run of the* **L***-automaton simulates a witness derivation in the corresponding* **L***-grammar,*

and vice versa, every witness derivation in an **L***-grammar is simulated by an accepting run of the corresponding* **L***-automaton.* In particular, such a tightly-related pair define the same language. Furthermore, if some word has multiple witness derivations, then it has multiple accepting runs of the corresponding **L**-automaton (each run simulating one derivation), and vice versa. We will show how to construct a tightly-related **L**-automaton M_G to a given the grammar G, and vice versa, how to construct tightly-related **L**-grammar G_M to a given **L**-automaton M.

4.1 The tight relation between **L***-automata and* **L***-grammars*

We start by defining how an accepting run of an **L**-automaton simulates a witness derivation in **L**.

DEFINITION 4.1
Let $G = (\Sigma, \mathcal{B}, \tau_0, \alpha)$ be an **L**-grammar, and $M = (\Sigma, \mathcal{O}, X_0, \delta)$ be an **L**-automaton. Let $T : \mathcal{O} \longrightarrow \mathcal{C}$ be a mapping[7] from $\mathcal{O}$ into categories over $\mathcal{B}$. Given a marked **L**-derivation tree Δ of length n ($n \geq 1$):

$$\Delta = \langle (v_1, \text{children}(v_1), \Gamma_1 \rhd \tau_1), \ldots, (v_n, \text{children}(v_n), \Gamma_n \rhd \tau_n) \rangle$$

and a run $\pi = c_0 \vdash^n c_n$ of M of length n, we say that π and Δ are ***T*-related** (denoted by $\boldsymbol{\pi \sim_T \Delta}$) iff for every $1 \leq j \leq n$ the following conditions hold:

1. $\Gamma_j \rhd \tau_j$ is the axiom $\tau_j \rhd \tau_j$ and $\text{children}(v_j) = \sigma$, where $\tau_j \in \alpha[\sigma]$, iff the move $c_{j-1} \vdash c_j$ is $shift\,[\sigma, X]$ and $\tau_j = T(X)$;
2. $\Gamma_j \rhd \tau_j$ is the axiom $[\tau_j] \rhd \tau_j$ (τ_j is a labeled assumption) iff the move $c_{j-1} \vdash c_j$ is $hypothesis - shift\,[X]$ and $\tau_j = T(X)$;
3. $\Gamma_j \rhd \tau_j$ is obtained from $\Gamma_m \rhd \tau_m$ ($m < j-1$) and $\Gamma_{j-1} \rhd \tau_{j-1}$ by an arrow elimination rule application iff the move $c_{j-1} \vdash c_j$ is $reduce\,[Y, Z, X]$ (where $T(Y) = \tau_m$, $T(Z) = \tau_{j-1}$) and $\tau_j = T(X)$;
4. $\Gamma_j \rhd \tau_j$ is obtained from $\Gamma_{j-1} \rhd \tau_{j-1}$ by an arrow introduction ($\rightarrow I$) rule application iff the move $c_{j-1} \vdash c_j$ is $left - release\,[X, Y, Z]$ (where $T(Z) = \tau_{j-1}$) and $\tau_j = T(X)$;
5. $\Gamma_j \rhd \tau_j$ is obtained from $\Gamma_{j-1} \rhd \tau_{j-1}$ by an arrow introduction ($\leftarrow I$) rule application iff the move $c_{j-1} \vdash c_j$ is $right - release\,[X, Y, Z]$ (where $T(Z) = \tau_{j-1}$) and $\tau_j = T(X)$.

Tight-relatedness is very strong in that it covers every derivation step and every automaton move, making correspondence between them. Thus, one immediate observation is that if $\pi \sim_T \Delta$, then so are the subsequences of π and Δ of the same length starting at the same index. Following is a demonstration of this correspondence for an arrow elimination rule application.

Let Δ be a single derivation tree, where the last inference rule is arrow elimination:

$$\Delta : \dfrac{\Delta_1 : \dfrac{\vdots}{\Gamma_m \rhd \tau_m} \quad \Delta_2 : \dfrac{\vdots}{\Gamma_{n-1} \rhd \tau_{n-1}}}{\Gamma_n \rhd \tau_n} \; (\rightarrow E)$$

[7] Note, that T is not required to be an one-to-one or onto function. Indeed, there is no one-to-one and onto function from a finite set of objects $\mathcal{O}$ to an infinite set of categories C.

or in a linearized notation:

$$\Delta = \begin{array}{l}\langle (v_1, \text{children}(v_1), \Gamma_1 \rhd \tau_1), \ldots, (v_m, \text{children}(v_m), \Gamma_m \rhd \tau_m), \ldots, \\ (v_{n-1}, \text{children}(v_{n-1}), \Gamma_{n-1} \rhd \tau_{n-1}), (v_n, \{v_m, v_{n-1}\}, \Gamma_n \rhd \tau_n)\rangle\end{array}$$

and let $\pi \sim_T \Delta$. Clearly, Δ may be partitioned into three parts:

1. The prefix of Δ, $\langle (v_1, \text{children}(v_1), \Gamma_1 \rhd \tau_1), \ldots, (v_m, \text{children}(v_m), \Gamma_m \rhd \tau_m)\rangle$, is a linearization of Δ_1;
2. The middle part of Δ,

   $$\langle (v_{m+1}, \text{children}(v_{m+1}), \Gamma_{m+1} \rhd \tau_{m+1}), \ldots, (v_{n-1}, \text{children}(v_{n-1}), \Gamma_{n-1} \rhd \tau_{n-1})\rangle\,,$$

   is a linearization of Δ_2;
3. The last part is the root of Δ, $\langle (v_n, \{v_m, v_{n-1}\}, \Gamma_n \rhd \tau_n)\rangle$.

Since the i-th derivation step of Δ corresponds to i-th automaton move in π, π may also be partitioned into three parts π_1, π_2 and π_3, where $\pi_1 \sim_T \Delta_1$, $\pi_2 \sim_T \Delta_2$ and $\pi_3 \sim_T \Delta_3$. As we will show in proposition 4.3, π_1 and π_2 are balanced runs, implying that:

- at the end of π_1 HS contains a single element (Y, β_1), where $\tau_m = T(Y)$;
- at the end of π_2 (preceded by π_1) HS contains two elements $(Y, \beta_1)(Z, \beta_2)$, where $\tau_{n-1} = T(Z)$.

Finally, after the last (*reduce*) step of π the elements $(Y, \beta_1)(Z, \beta_2)$ are popped from HS and a new element (X, γ) is pushed onto HS, where $\tau_n = T(X)$.

Thus we arrive at an important observation, that after a sequence of transitions $\pi_1 \cdot \pi_2$, corresponding to a linearization of two subtrees Δ_1, Δ_2, HS contains two elements, whose main objects Y, Z correspond to the roots of the subtrees Δ_1, Δ_2. A similar statement is true for arrow introduction rule applications in Δ. This may be further generalized, stating that after a run T-related to an ordered **forest** of derivation trees Δ, HS contains elements with main objects corresponding to the roots of Δ.

The following observation is slightly more complicated; it is summarized in the proposition below. In the following proposition we use $input_\pi(i, j)$ to denote the input that is read during the subrun $c_i \vdash^+ c_j$ of π.

DEFINITION 4.2
Let $\pi = c_0 \vdash^n c_n$ be a run of an **L**-automaton and let $c_i \vdash^+ c_j$ be a subrun of π, containing the following shift transitions:
shift $[\sigma_1, X_1], \ldots,$ shift $[\sigma_m, X_m]$ in the order of their appearance in $c_i \vdash^+ c_j$.
Then, $\mathbf{input_\pi(i, j)} = \sigma_1 \ldots \sigma_m$.

In the next proposition we naturally extend T to sequences of objects from $\mathcal{O}$:

$$T(\epsilon) = \epsilon, \quad T(X_1 \ldots X_n) = T(X_1) \ldots T(X_n)$$

PROPOSITION 4.3
Let $M = (\Sigma, \mathcal{O}, X_0, \delta)$ be an **L**-automaton and $G = (\Sigma, \mathcal{B}, \tau_0, \alpha)$ be an **L**-grammar, and let $T : \mathcal{O} \longrightarrow \mathcal{C}$ be a mapping from the $\mathcal{O}$ into the categories over B. Let Δ be a marked derivation tree

$$\langle (v_1, \text{children}(v_1), \Gamma_1 \rhd \tau_1), \ldots, (v_k, \text{children}(v_k), \Gamma_k \rhd \tau_k)\rangle$$

and $\pi = \langle w, \nu\rangle \vdash \langle w_1, \nu_1\rangle \vdash^{k-1} \langle w_k, \nu_k\rangle$ be a run of M, such that $\pi \sim_T \Delta$: Then the following conditions hold:

1. π is a balanced run: $\pi = \langle w, \nu\rangle \vdash^k \langle w_k, \nu \cdot (X_k, \beta_k)\rangle$
2. $\mathrm{yield}(\Delta) = \mathrm{input}_\pi(0, k)$;
3. • $\Gamma_k = [\Gamma_k^l]\hat{\Gamma}_k'[\Gamma_k^r]$ $(\hat{\Gamma}_k' \neq \epsilon)$ and $\beta_k = \beta_k^l \star \beta_k^r$, where $\Gamma_k^l = T(\beta_k^l)$ and $\Gamma_k^r = T(\beta_k^r)$; or
   • $\Gamma_k = [\Gamma_k']$ and $\Gamma_k' = T(\beta_k)$;

Proof. We prove the proposition by induction on k, the length of Δ and π.
$k = 1$: Δ is the axiom $\Gamma_1 \rhd \tau_1$, and π contains one step $\langle w, \nu\rangle \vdash \langle w_1, \nu \cdot (X_1, \beta_1)\rangle$.

1. during π the single element (X_1, β_1) is added onto HS; thus π is certainly a balanced run.
2. • If the first move is $hypothesis - shift\,[X_1]$ then $\mathrm{input}_\pi(0, 1) = \epsilon$ and by definition 4.1 $\Gamma_1 = [\tau_1]$, implying that $\mathrm{yield}(\Delta) = \epsilon$;
   • if the first move $shift\,[\sigma, X_1]$ then $\mathrm{input}_\pi(0, 1) = \sigma$ and by definition 4.1 $\Gamma_1 = \tau_1$ and $\mathrm{children}(v_1) = \sigma$ where $\tau_1 \in \alpha[\sigma]$, implying that $\mathrm{yield}(\Delta) = \sigma$.
3. • if $\Gamma_1 = \tau_1$ then by definition 4.1 the first move is $shift$, implying that $\beta_1 = \star$;
   • if $\Gamma_1 = [\tau_1]$ then by definition 4.1 the first move is $hypothesis - shift$, implying that $\beta_1 = X_1$, where $\tau_1 = T(X_1)$.

Suppose the conditions hold for Δ and π of length$\leq k$ $(k \geq 1)$, and let Δ and π be of length $k + 1$:

$$\Delta = \langle (v_1, \mathrm{children}(v_1), \Gamma_1 \rhd \tau_1), \ldots, (v_{k+1}, \mathrm{children}(v_{k+1}), \Gamma_{k+1} \rhd \tau_{k+1})\rangle$$

$$\pi = \langle w, \nu\rangle \vdash \langle w_1, \nu_1\rangle \vdash^k \langle w_{k+1}, \nu_{k+1}\rangle$$

There are five possibilities for the last move of π and the last inference rule application of Δ, respectively:

- shift + axiom:
  This is impossible for the trees of length≥ 2, because the only derivation tree that ends with the axiom is the axiom itself.
- hypothesis-shift + axiom:
  Again, this case is impossible.
- reduce + arrow elimination:
  The k'th move is

  $$\langle w_k, \nu_k \cdot (Y, \beta_Y)(Z, \beta_Z)\rangle \vdash^{\mathrm{reduce}} \langle w_k, \nu_k \cdot (X_{k+1}, \beta_{k+1})\rangle$$

  and Δ is obtained from two subtrees Δ_1 and Δ_2 by an arrow elimination rule application:

  $$\Delta: \quad \frac{\Delta_1: \begin{array}{c}\vdots\\ \Gamma_i \rhd \tau_i\end{array} \quad \Delta_2: \begin{array}{c}\vdots\\ \Gamma_k \rhd \tau_k\end{array}}{\Gamma_{k+1} \rhd \tau_{k+1}}$$

  where $\tau_{k+1} = T(X_{k+1})$. Note, that $\mathrm{yield}(\Delta) = \mathrm{yield}(\Delta_1) \cdot \mathrm{yield}(\Delta_2)$.
  By the T-relation definition, as explained in page 21, π can be partitioned into

three parts, where the first part $\pi_1 \sim_T \Delta_1$, the middle part $\pi_2 \sim_T \Delta_2$, and the last part π_3 is a single reduce step:

$$\begin{aligned}\pi_1 &= \langle w, \nu\rangle \vdash^i \langle w_i, \nu_i \cdot (Y, \beta_Y)\rangle\\ \pi_2 &= \langle w_i, \nu_i \cdot (Y, \beta_Y)\rangle \vdash^{k-i} \langle w_k, \nu_k \cdot (Y, \beta_Y)(Z, \beta_Z)\rangle\\ \pi_3 &= \langle w_k, \nu_k \cdot (Y, \beta_Y)(Z, \beta_Z)\rangle \vdash^{\text{reduce}} \langle w_k, \nu_k \cdot (X_{k+1}, \beta_{k+1})\rangle\end{aligned}$$

where $\tau_i = T(Y)$ and $\tau_k = T(Z)$.

1. By the induction hypothesis π_1 and π_2 are balanced runs, i.e. $\nu_i = \nu_k = \nu$, where $c_j = \langle w_j, \nu\nu'_j\rangle$ $(1 \le j \le k)$. Therefore, $\pi = \langle w, \nu\rangle \vdash^{k+1} \langle w_{k+1}, \nu \cdot (X_{k+1}, \beta_{k+1})\rangle$ is a balanced run too.
2. First, note that $\text{input}_\pi(0, k+1) = \text{input}_\pi(0, i) \cdot \text{input}_\pi(i+1, k)$, because the last move of π is not *shift*.
   Let $\text{input}_\pi(0, i) = u_1$ and $\text{input}_\pi(i+1, k) = u_2$ $(u_1, u_2 \in \Sigma^\star)$, where $\text{input}_\pi(0, k+1) = u_1u_2$. Then by the induction hypothesis $\text{yield}(\Delta_1) = u_1$ and $\text{yield}(\Delta_2) = u_2$, thus implying that $\text{yield}(\Delta) = \text{yield}(\Delta_1) \cdot \text{yield}(\Delta_2) = u_1u_2$.
3. Since Δ is a marked derivation tree, the labeled assumptions of Γ_{k+1} are peripheral. Therefore, Γ_{k+1} has one of the following forms:
   $\Gamma_{k+1} = [\Gamma^l_{k+1}]\hat{\Gamma}'_{k+1}[\Gamma^r_{k+1}]$ $(\Gamma'_{k+1} \neq \epsilon)$ or $\Gamma_{k+1} = [\Gamma'_{k+1}]$.
   - If $\Gamma_{k+1} = [\Gamma^l_{k+1}]\hat{\Gamma}'_{k+1}[\Gamma^r_{k+1}]$ $(\Gamma'_{k+1} \neq \epsilon)$
     then by the arrow elimination rule definition there are three possibilities:
     (a) if $\Gamma_i = [\Gamma^l_i]\hat{\Gamma}'_i[\Gamma^r_i]$ $(\Gamma'_i \neq \epsilon)$ and $\Gamma_k = [\Gamma'_k]$, where $\Gamma_{k+1} = [\Gamma^l_i]\hat{\Gamma}'_i[\Gamma^r_i\Gamma'_k]$,
     then by the induction hypothesis $\beta_Y = \beta^l_Y \star \beta^r_Y$ (where $\Gamma^l_i = T(\beta^l_Y)$, $\Gamma^r_i = T(\beta^r_Y)$) and $\Gamma'_k = T(\beta_Z)$, which by the definition of *reduce* implies that $\beta_{k+1} = \beta^l_Y \star \beta^r_Y\beta_Z$, where $\Gamma^l_i = T(\beta^l_Y)$ and $\Gamma^r_i\Gamma'_k = T(\beta^r_Y\beta_Z)$.
     (b) $\Gamma_i = [\Gamma'_i]$ and $\Gamma_k = [\Gamma^l_k]\hat{\Gamma}'_k[\Gamma^r_k]$ $(\Gamma'_k \neq \epsilon)$, where $\Gamma_{k+1} = [\Gamma'_i\Gamma^l_k]\hat{\Gamma}'_k[\Gamma^r_k]$.
     This case is symmetric to the previous case.
     (c) if $\Gamma_i = [\Gamma^l_i]\hat{\Gamma}'_i$ $(\Gamma'_i \neq \epsilon)$ and $\Gamma_k = \hat{\Gamma}'_k[\Gamma^r_k]$ $(\Gamma'_k \neq \epsilon)$, where $\Gamma_{k+1} = [\Gamma^l_i]\hat{\Gamma}'_i\hat{\Gamma}'_k[\Gamma^r_k]$,
     then by the induction hypothesis $\beta_Y = \beta^l_Y\star$ (where $\Gamma^l_i = T(\beta^l_Y)$) and $\beta_Z = \star\beta^r_Z$ (where $\Gamma^r_k = T(\beta^r_Z)$), which by the definition of *reduce* implies that $\beta_{k+1} = \beta^l_Y \star \beta^r_Z$, where $\Gamma^l_i = T(\beta^l_Y)$ and $\Gamma^r_k = T(\beta^r_Z)$.
   - If $\Gamma_{k+1} = [\Gamma'_{k+1}]$ then by the arrow elimination rule definition $\Gamma_i = [\Gamma'_i]$ and $\Gamma_k = [\Gamma'_k]$, where $\Gamma_{k+1} = [\Gamma'_i\Gamma'_k]$,
     then by the induction hypothesis $\Gamma'_i = T(\beta_Y)$ and $\Gamma'_k = T(\beta_Z)$,
     which by the *reduce* step definition implies that

$$\Gamma_{k+1} = [\Gamma'_i\Gamma'_k] = T(\beta_Y\beta_Z) = T(\beta_{k+1})$$

- <u>left-release + arrow introduction:</u>
  The move $c_k \vdash c_{k+1}$ is

$$\langle w_k, \nu_k \cdot (X_k, Y\beta)\rangle \vdash^{\text{left-release}} \langle w_k, \nu_k \cdot (X_{k+1}, \beta)\rangle$$

and Δ is obtained from the subtree Δ_1 by an arrow introduction $(\to I)$ rule application:

$$\Delta: \quad \frac{\Delta_1: \begin{array}{c}\vdots\\ \overline{[\tau]\Gamma \rhd \tau_k}\end{array}}{\Gamma_{k+1} \rhd \tau_{k+1}}\ (\to I) \quad (\Gamma \neq \epsilon)$$

where $\tau_{k+1} = (\tau \to \tau_k) = T(X_{k+1})$ and $\Gamma_{k+1} = \Gamma$. Note, that $\text{yield}(\Delta) = \text{yield}(\Delta_1)$, because the discharged assumption $[\tau]$ does not add a letter to $\text{yield}(\Delta_1)$. Therefore, π may be partitioned into two parts, where the first part $\pi_1 \sim_T \Delta_1$, and the second part π_2 is the single left-release step:

$$\begin{aligned} \pi_1 &= \langle w, \nu \rangle \vdash^k \langle w_k, \nu_k \cdot (X_k, Y\beta) \rangle \\ \pi_2 &= \langle w_k, \nu_k \cdot (X_k, Y\beta) \rangle \vdash^{\text{left-release}} \langle w_k, \nu_k \cdot (X_{k+1}, \beta) \rangle \end{aligned}$$

where $\tau_k = T(X_k)$.

1. By the induction hypothesis π_1 is a balanced run, i.e. $\nu_k = \nu$, where $c_j = \langle w_j, \nu\nu'_j \rangle$ $(1 \leq j \leq k)$. Therefore, $\pi = \langle w, \nu \rangle \vdash^{k+1} \langle w_k, \nu \cdot (X_{k+1}, \beta) \rangle$ is a balanced run too.
2. First, note that $\text{input}_\pi(0, k+1) = \text{input}_\pi(0, k)$, because the last move of π is not *shift*. Let $\text{input}_\pi(0, k) = \text{input}_\pi(0, k+1) = u$ $(u \in \Sigma^\star)$. Then by the induction hypothesis $\text{yield}(\Delta_1) = u$, thus implying that $\text{yield}(\Delta) = \text{yield}(\Delta_1) = u$.
3. Since Δ is a marked derivation tree, the labeled assumptions of Γ_{k+1} are peripheral. Therefore, Γ_{k+1} has one of the following forms:
   $\Gamma_{k+1} = [\Gamma^l_{k+1}]\hat{\Gamma}'_{k+1}[\Gamma^r_{k+1}]$ $(\Gamma'_{k+1} \neq \epsilon)$ or $\Gamma_{k+1} = [\Gamma'_{k+1}]$.
   - If $\Gamma_{k+1} = [\Gamma^l_{k+1}]\hat{\Gamma}'_{k+1}[\Gamma^r_{k+1}]$ $(\Gamma'_{k+1} \neq \epsilon)$,
     then by the arrow introduction rule definition $[\tau]\Gamma = [\tau\Gamma^l_{k+1}]\hat{\Gamma}'_{k+1}[\Gamma^r_{k+1}]$,
     implying by the induction hypothesis that $Y\beta = Y\beta^l \star \beta^r$, where $\tau\Gamma^l_{k+1} = T(Y\beta^l)$, $\Gamma^r_{k+1} = T(\beta^r)$, and thus $\beta = \beta^l \star \beta^r$, where $\Gamma^l_{k+1} = T(\beta^l)$, $\Gamma^r_{k+1} = T(\beta^r)$.
   - If $\Gamma_{k+1} = [\Gamma'_{k+1}]$,
     then by the arrow introduction rule definition $[\tau]\Gamma = [\tau\Gamma'_{k+1}]$ $(\Gamma'_{k+1} \neq \epsilon)$,
     implying by the induction hypothesis that $\tau\Gamma'_{k+1} = T(Y\beta)$, and thus $\Gamma'_{k+1} = T(\beta)$.

- right-release + arrow introduction:
  This case is symmetric to the previous case (left-release + arrow introduction).

■

Now we are ready to formulate the relationships between the **L**-automaton and the **L**-grammar.

DEFINITION 4.4
(Automaton-grammar tight relation)
Let $G = (\Sigma, \mathcal{B}, \tau_0, \alpha)$ be an **L**-grammar, and let $M = (\Sigma, \mathcal{O}, X_0, \delta)$ be an **L**-automaton. We say that G and M are **tightly related** (denoted by $\boldsymbol{M \sim G}$) iff there is a mapping $T : \mathcal{O} \longrightarrow \mathcal{C}$ such that $\tau_0 = T(X_0)$ and:

- for every marked normal witness derivation Δ in G there is a run $\pi = c_{ini} \vdash^n c_n$ of M such that $\pi \sim_T \Delta$;
- for every accepting run $\pi = c_{ini} \vdash^n c_{acc}$ of M there is a marked **L**-derivation Δ such that $\pi \sim_T \Delta$.

Note that since T is a function, then there is a **unique** marked **L**-derivation Δ for every accepting run π in the above definition. On the other hand, since T is not required to be a one-to-one function, then there may be several runs for some marked normal witness derivations in the above definition.

EXAMPLE 4.5
The automaton M_1 from example 3.3 and the grammar G_2 from example 2.8 are tightly related ($M_1 \sim G_2$). A mapping T between $\mathcal{O}$ and $\mathcal{C}$ is:

$$\begin{aligned}
T(X_0) &= A \\
T(X_1) &= B \\
T(X_2) &= ((B \to C) \leftarrow B) \\
T(X_3) &= (B \to C) \\
T(X_4) &= C \\
T(X_5) &= (C \to A) \\
T(X_6) &= (B \to A) \\
T(X_7) &= ((B \to A) \to (B \to C)) \\
T(X_8) &= B
\end{aligned}$$

We do not prove here that $M_1 \sim G_2$. We will reach this conclusion when demonstrating how the **L**-automaton may be constructed from the **L**-grammar, and vice versa. In this example we show that the run π_1 of M_1 from Figure 8 and the derivation tree Δ_1 of G_2 from example 2.8 are T-related ($\pi_1 \sim_T \Delta_1$). Figure 9 demonstrates this.

Note that the mapping T in the tight relation is not unique. Namely, one may find an **L**-automaton M and an **L**- grammar G, where there are two different mappings T and T' between $\mathcal{O}$ and C such that both T and T' satisfy the conditions as required by definition 4.4. This is demonstrated in the following example.

EXAMPLE 4.6
Let $G = (\{a, b\}, (A, B, C), \alpha, A)$ be an **L**-grammar, where

$$\alpha[a] = \{B, C\} \text{ and } \alpha[b] = \{(B \to A), (C \to A)\}$$

and let $M = (\{a, b\}, \{X_0, X_1, X_2, X_3, X_4\}, X_0, \delta)$ be an **L**-automaton, where δ contains the following transitions:

- shift$[a, X_1]$, shift$[a, X_3]$, shift$[b, X_2]$, shift$[b, X_4]$;
- hypothesis-shift$[X_i]$ for every $0 \leq i \leq 4$;
- reduce$[X_1, X_2, X_0]$, reduce$[X_3, X_4, X_0]$;
- left-release$[X_2, X_1, X_0]$, left-release$[X_4, X_3, X_0]$.

Consider the following mappings T and T' between $\mathcal{O}$ and C:

$$T(X_0) = A,\ T(X_1) = B,\ T(X_2) = (B \to A),\ T(X_3) = C,\ T(X_4) = (C \to A)$$

$$T'(X_0) = A,\ T'(X_1) = C,\ T'(X_2) = (C \to A),\ T'(X_3) = B,\ T'(X_4) = (B \to A)$$

We show that both T and T' satisfy the conditions as required by definition 4.4. Note, that $L(G) = \{ab\}$, where the only witness derivations in G are:

$$\Delta_1: \dfrac{\dfrac{\dfrac{a}{B} \quad \dfrac{b}{(B \to A)}}{A}\ (\to E)}{} \qquad \Delta_2: \dfrac{\dfrac{\dfrac{a}{C} \quad \dfrac{b}{(C \to A)}}{A}\ (\to E)}{}$$

and $L(M) = \{ab\}$, where the only accepting runs are:

$$\pi_1 = \langle ab, \epsilon\rangle \vdash \langle b, (X_1, \star)\rangle \vdash \langle \epsilon, (X_1, \star)(X_2, \star)\rangle \vdash \langle \epsilon, (X_0, \star)\rangle$$

$$
\begin{array}{l}
\pi_1 = \\
\qquad \underline{\Delta_1 = \langle} \\
\langle ababab, \epsilon\rangle \\
\qquad \underline{(v_{10}, \epsilon, B \rhd B),} \\
\vdash^{\text{shift}} \langle babab, (X_8, \star)\rangle \qquad\qquad B = T(X_8) \\
\qquad \underline{(v_3, \epsilon, [B] \rhd B),} \\
\vdash^{\text{hypothesis-shift}} \langle babab, (X_8, \star)(X_1, X_1)\rangle \qquad\qquad B = T(X_1) \\
\qquad \underline{(v_1, \epsilon, ((B \to C) \leftarrow B) \rhd ((B \to C) \leftarrow B)),} \\
\vdash^{\text{shift}} \langle abab, (X_8, \star)(X_1, X_1)(X_2, \star)\rangle \qquad\qquad ((B \to C) \leftarrow B) = T(X_2) \\
\qquad \underline{(v_2, \epsilon, B \rhd B),} \\
\vdash^{\text{shift}} \langle bab, (X_8, \star)(X_1, X_1)(X_2, \star)(X_8, \star)\rangle \qquad\qquad B = T(X_8) \\
\qquad \underline{(v_4, (v_1, v_2), ((B \to C) \leftarrow B)\ B \rhd (B \to C)),} \\
\vdash^{\text{reduce}} \langle bab, (X_8, \star)(X_1, X_1)(X_3, \star)\rangle \qquad\qquad (B \to C) = T(X_3) \\
\qquad \underline{(v_5, (v_3, v_4), [B]\ ((B \to C) \leftarrow B)\ B \rhd C),} \\
\vdash^{\text{reduce}} \langle bab, (X_8, \star)(X_4, X_1\star)\rangle \qquad\qquad C = T(X_4) \\
\qquad \underline{(v_6, \epsilon, (C \to A) \rhd (C \to A)),} \\
\vdash^{\text{shift}} \langle ab, (X_8, \star)(X_4, X_1\star)(X_5, \star)\rangle \qquad\qquad (C \to A) = T(X_5) \\
\qquad \underline{(v_7, (v_5, v_6), [B]\ ((B \to C) \leftarrow B)\ B\ (C \to A) \rhd A),} \\
\vdash^{\text{reduce}} \langle ab, (X_8, \star)(X_0, X_1\star)\rangle \qquad\qquad A = T(X_0) \\
\qquad \underline{(v_8, (v_7),), ((B \to C) \leftarrow B)\ B\ (C \to A) \rhd (B \to A)),} \\
\vdash^{\text{left-release}} \langle ab, (X_8, \star)(X_6, \star)\rangle \qquad\qquad (B \to A) = T(X_6) \\
\qquad \underline{(v_9, \epsilon, ((B \to A) \to (B \to C)) \rhd ((B \to A) \to (B \to C))),} \\
\vdash^{\text{shift}} \langle b, (X_8, \star)(X_6, \star)(X_7, \star)\rangle \qquad\qquad ((B \to A) \to (B \to C)) = T(X_7) \\
\qquad \underline{(v_{11}, (v_8, v_9), ((B \to C) \leftarrow B)\ B\ (C \to A)\ ((B \to A) \to (B \to C)) \rhd (B \to C)),} \\
\vdash^{\text{reduce}} \langle b, (X_8, \star)(X_3, \star)\rangle \qquad\qquad (B \to C) = T(X_3) \\
\qquad \underline{(v_{12}, (v_{10}, v_{11}), B\ ((B \to C) \leftarrow B)\ B\ (C \to A)\ ((B \to A) \to (B \to C)) \rhd C),} \\
\vdash^{\text{reduce}} \langle b, (X_4, \star)\rangle \qquad\qquad C = T(X_4) \\
\qquad \underline{(v_{13}, \epsilon, (C \to A) \rhd (C \to A)),} \\
\vdash^{\text{shift}} \langle \epsilon, (X_4, \star)(X_5, \star)\rangle \qquad\qquad (C \to A) = T(X_5) \\
\underline{(v_{14}, (v_{12}, v_{13}), B\ ((B \to C) \leftarrow B)\ B\ (C \to A)\ ((B \to A) \to (B \to C))\ (C \to A) \rhd A)\rangle} \\
\vdash^{\text{reduce}} \langle \epsilon, (X_0, \star)\rangle \qquad\qquad A = T(X_0)
\end{array}
$$

FIG. 9. The run π_1 and the derivation tree Δ_1 are T-related

$$\pi_2 = \langle ab, \epsilon\rangle \vdash \langle b, (X_3, \star)\rangle \vdash \langle \epsilon, (X_3, \star)(X_4, \star)\rangle \vdash \langle \epsilon, (X_0, \star)\rangle$$

When considering the mapping T, it is easy to see that $\pi_1 \sim_T \Delta_1$ and $\pi_2 \sim_T \Delta_2$; similarly, when considering the mapping T', it is obvious that $\pi_2 \sim_{T'} \Delta_1$ and $\pi_1 \sim_{T'} \Delta_2$.

The non-surprising conclusion of an automaton and a grammar being tightly related is that they accept the same languages.

PROPOSITION 4.7
Let M be an **L**-automaton and G be an **L**-grammar.
Then, $M \sim G$ implies $L(M) = L(G)$.

PROOF.

- $L(M) \subseteq L(G)$:

Let $w \in L(M)$, $w \neq \epsilon$, i.e. M has an accepting run $\pi = c_{ini} \vdash^n c_{acc}$

$$\pi = \langle w, \epsilon \rangle \vdash^+ \langle \epsilon, (X_0, \star) \rangle$$

Since $M \sim G$, there is a marked derivation tree Δ such that $\pi \sim_T \Delta$.
Let $\text{root}(\Delta) = \Gamma \rhd \tau$; then, $\tau = T(X_0) = \tau_0$. Furthermore, by proposition 4.3 (condition 2), $\text{yield}(\Delta) = \text{input}_\pi(0, n) = w$. Since $\beta_n = \star$, then by proposition 4.3 (condition 3) Γ does not contain labeled assumptions, i.e. $\Gamma \in \alpha[\text{yield}(\Delta)]$, thus implying that $\Gamma \in \alpha[w]$. Therefore, Δ is the witness derivation for w.

- $L(G) \subseteq L(M)$:
  Let $w \in L(G)$, $w \neq \epsilon$, i.e. there is a (normal) witness derivation $\Gamma \rhd \tau_0$ in G, where $\Gamma \in \alpha[w]$.

  $$\Delta = \langle (v_1, \text{children}(v_1), \Gamma_1 \rhd \tau_1), \ldots, (v_n, \text{children}(v_n), \Gamma_n \rhd \tau_n) \rangle$$

  where $\Gamma_n = \Gamma$ and $\tau_n = \tau_0$.
  Since $M \sim G$, M has a run $\pi = c_{ini} \vdash^n +c_n$ such that $\pi \sim_T \Delta$; furthermore, π is a balanced run by proposition 4.3 (condition 1), which implies that $c_n = \langle \epsilon, (X_0, \beta_n) \rangle$. Since $\Gamma_n = \Gamma$ does not contain labeled assumptions then by proposition 4.3 (condition 3) $\beta_n = \star$ and thus

  $$c_n = \langle \epsilon, (X_0, \star) \rangle = c_{acc}$$

  implying that π is an accepting run.
  By proposition 4.3 (condition 2), $\text{input}_\pi(0, n) = \text{yield}(\Delta) = w$, i.e, M reads the input word w during the run π; $w \in L(M)$ follows.

∎

4.2 **L**-*automaton constraints*

The **L**-automaton was designed as a computational tool tightly related to **L**-grammars by simulating Natural Deduction derivations. As we showed in the previous section, there is a remarkable correspondence between *reduce* transitions and arrow elimination rules, *release* transitions and arrow introduction rules, $shift$ transitions and axioms. However, this correspondence is violated by the following fact: in an arbitrary **L**-automaton there are no dependencies between different kinds of transitions, whereas in **L** there are implicit constraints on categories appearing in inference rule. For example, because of constraints 4, 2 and 5, it is impossible to have an **L**-automaton containing both of the two following *reduce* transitions:

$$\textit{reduce}\,[X_1, X_2, X]\,,\ \textit{reduce}\,[X_2, X_1, X]$$

which would have required the following impossible situation in the grammar:

$$\frac{\tau_1 \quad \tau_2}{\tau} \qquad \frac{\tau_2 \quad \tau_1}{\tau}$$

where $T(X_1) = \tau_1$, $T(X_2) = \tau_2$ and $T(X) = \tau$. Another example is that in an arbitrary **L**-derivation we may assume any category and further discharge it. However,

an **L**-automaton may put in an h-list only those objects, for which $hypothesis-shift$ transition is defined. This motivates to constrain the transition relation δ of an **L**-automaton. These constraints, induced by inference rules of **L**, make every **L**-automaton being tightly related to some **L**-grammar, and vice versa.

Let us review the constraints appearing in definition 3.1 and explain them. Let M be an **L**-automaton and let G be an **L**-grammar, such that $M \sim G$; let T be the mapping from $\mathcal{O}$ into $\mathcal{C}$ as required by definition 4.4.

First, we explain the reason for using the equivalence relation $R_{\approx}$ induced by M. Let $left{-}release\,[X, Y, Z]$ and $left{-}release\,[X_1, Y_1, Z_1]$ for some $X, Y, Z, X_1, Y_1, Z_1 \in \mathcal{O}$. $M \sim G$ implies that $T(X) = (T(Y) \to T(Z))$ and $T(X_1) = (T(Y_1) \to T(Z_1))$. Suppose now that $Y = Y_1$ and $Z = Z_1$. Then, $T(X) = (T(Y) \to T(Z)) = (T(Y_1) \to T(Z_1)) = T(X_1)$. The other direction is similar: suppose $X = X_1$. Then, $T(X) = T(X_1)$ implies $(T(Y) \to T(Z)) = (T(Y_1) \to T(Z_1))$, and thus $T(Y) = T(Y_1)$ and $T(Z) = T(Z_1)$. To sum up, we obtained that $T(X) = T(X_1)$ iff $T(Y) = T(Y_1)$ and $T(Z) = T(Z_1)$. Thus, the equivalence relation $R_{\approx}$ actually contains pairs of objects $(U, W) \in R_{\approx}$ such that $T(U) = T(Z)$.

We extend T to equivalence classes as follows: $T([X]) = T(X)$. Since $T(X_1) = T(X_2)$ for every $X_1, X_2 \in [X]$, then $T([X])$ is well defined.

Let $R_{\approx}$ be the equivalence relation induced by M and let Π be the set of all the equivalence classes of $R_{\approx}$. Below we give some intuition for the constraints on automaton transitions, specified in the definition 3.1. As we will show later in this section, these constraints are necessary for building an **L**-grammar tightly related to a given **L**-automaton.

1. For every $X \in \Pi$ there is $X_1 \in [X]$ such that $hypothesis - shift\,[X_1]$.
   As we noted above, in an arbitrary **L**-grammar any category may be assumed as a hypothesis. Likewise, an **L**-automaton should have a $hypothesis{-}shift$ transition for at least one representative of every equivalence class.
2. For all $X_1, Y_1, Z_1, X_2, Y_2, Z_2 \in \mathcal{O}$:
   left-release$[X_1, Y_1, Z_1]$ and right-release$[X_2, Y_2, Z_2]$ imply $[X_1] \neq [X_2]$.
   This constraint actually says that two categories cannot be equal if one of them is $(T(Y_1) \to T(Z_1))$, while the other is $(T(Y_2) \leftarrow T(Z_2))$.
3. For all $X_1, Y_1, Z_1 \in \mathcal{O}, X_2 \in [X_1], Y_2 \in [Y_1], Z_2 \in [Z_1]$:
   $left{-}release\,[X_1, Y_1, Z_1]$ iff $left{-}release\,[X_2, Y_2, Z_2]$, whereas $right{-}release\,[X_1, Y_1, Z_1]$ iff $right - release\,[X_2, Y_2, Z_2]$.
   This constraint avoids the following problem. Let M be the following automaton: $M = (\{a, b, c\}, \{X, Y, Z_1, Z_2, P, Q, U, V\}, Q, \delta)$, where δ contains the following transitions:
   - $shift\,[a, X]$, $shift\,[b, Y]$, $shift\,[c, P]$;
   - $hypothesis{-}shift\,[X]$, $hypothesis{-}shift\,[Y]$, $hypothesis{-}shift\,[Z_1]$, $hypothesis{-}shift\,[Z_2]$,
     $hypothesis{-}shift\,[P]$, $hypothesis{-}shift\,[Q]$, $hypothesis{-}shift\,[U]$, $hypothesis{-}shift\,[V]$;
   - $reduce\,[X, Y, Z_1]$, $left - release\,[Y, X, Z_1]$;
   - $reduce\,[Z_2, P, Q]$, $left - release\,[P, Z_2, Q]$;
   - $reduce\,[Z_1, U, V]$, $right - release\,[Z_1, U, V]$;
   - $reduce\,[Z_2, U, V]$, $right - release\,[Z_2, U, V]$;

Note, that M satisfies all the constraints except constraint 3. If there is an **L**-grammar G s.t. $M \sim G$, then, as we mentioned earlier:
• $T(Y) = (T(X) \to T(Z_1))$
• $T(P) = (T(Z_2) \to T(Q))$
• $T(Z_1) = T(Z_2) = (T(U) \leftarrow T(V))$
Consider now the following witness derivation Δ for abc in G:

$$\Delta: \quad \dfrac{\dfrac{\dfrac{a}{T(X)} \quad \dfrac{b}{(T(X) \to T(Z_1))}}{T(Z_1)}\ (\to E) \quad \dfrac{c}{(T(Z_1) \to T(Q))}}{T(Q)}\ (\to E)$$

The first $(\to E)$ application in the above derivation may be simulated by $reduce\,[X, Y, Z_1]$:

$$\dfrac{T(X) \quad (T(X) \to T(Z_1))}{T(Z_1)}\ (\to E) \iff \text{reduce}\,[X, Y, Z_1]$$

and the second one may be simulated by $reduce\,[Z_2, P, Q]$:

$$\dfrac{T(Z_2) \quad (T(Z_2) \to T(Q))}{T(Q)}\ (\to E) \iff \text{reduce}\,[Z_2, P, Q]$$

Unfortunately, $Z_1 \neq Z_2$, and thus these two transitions cannot occur in successive automaton moves; furthermore, there is no accepting run of M which is T-related to Δ.
On the other hand, if M satisfied constraint 3, then δ would contain $reduce\,[Z_1, P, Q]$, which, applied after $reduce\,[X, Y, Z_1]$, would complete an accepting run of M T-related to Δ.

4. For all $X, Y, Z \in \mathcal{O}$:
$reduce\,[X, Y, Z]$ iff either $left - release\,[Y, X, Z]$ or $right - release\,[X, Y, Z]$.
This constraint simulates the following property of **L**-grammars: the arrow introduction rule
$$\dfrac{T(X)\Gamma \rhd T(Z)}{T(Y)}\ (\to I)$$
may be applied iff the arrow elimination rule
$$\dfrac{T(X) \quad T(Y)}{T(Z)}\ (\to E)$$
may be applied (which means that $T(Y) = (T(X) \to T(Z))$),
and similarly for $(\leftarrow I)$, $(\leftarrow E)$.
5. $D_M \cap I_\Pi = \emptyset$.
This constraint actually says that there are no cycles in dependency relation, i.e. it cannot happen that $[X_1]$ depends on $[X_2]$, $[X_2]$ depends on $[X_3]$,..., $[X_n]$ depends on $[X_1]$. Note that $([X_1], [X_2]) \in D_M$ means that $T([X_2]) \in ST(\{T[X_1]\})\backslash\{T[X_1]\}$. Thus, it is obvious that $T([X]) \notin ST(\{T[X]\})\backslash\{T[X]\}$ for any $[X] \in \Pi$.

4.3 Constructing an **L***-automaton from an* **L***-grammar*

In this section we show how to construct an **L**-automaton tightly related to a given **L**-grammar. Since only accepting runs are considered in the tight relation definition, then there are infinitely many **L**-automata tightly related to a given **L**-grammar. Indeed, a given **L**-automaton may be 'blown up' by adding redundant objects to the set $\mathcal{O}$ without adding *reduce* transitions with those objects; this will result in a new automaton having the same accepting runs as the original one. In this section we construct one of the infinitely many **L**-automata tightly related to a given **L**-grammar.

We want to emphasize that given an **L**-grammar G, there is a difference between a minimal automaton accepting the language $L(G)$ and a minimal automaton tightly related to G. Let us measure a minimality of an automaton as a number of objects in the set $\mathcal{O}$. In the constructed automaton a number of objects in $\mathcal{O}$ is $|ST(\{\tau_0\} \cup \bigcup_{\sigma\in\Sigma} \alpha[\sigma])|$; hence, every object is mapped to a *different* category, thus avoiding redundant objects that behave the same. This renders the constructed automaton a minimal automaton tightly related to G. However, G itself is not necessarily a minimal grammar accepting $L(G)$ (when measuring the minimality of a grammar as a size of its lexicon α). For example, a grammar G from example 4.6 is not a minimal grammar accepting the language $\{ab\}$. A grammar G' obtained from G by removing from its lexicon a basic category C (and all C-categories) accepts the same language $\{ab\}$, while having a smaller lexicon than G has. Clearly, an automaton M_G constructed from a grammar G is not a minimal automaton accepting the language $\{ab\}$; an automaton $M_{G'}$ constructed from a grammar G' is smaller than M_G, because it does not have objects mapped to categories C and $(C \to A)$.

To sum up, the constructed automaton is minimal among automata tightly related to a given **L**-grammar, but it is not minimal among automata accepting a given language.

THEOREM 4.8
For every **L**-grammar $G = (\Sigma, \mathcal{B}, \tau_0, \alpha)$ there is an **L**-automaton $M_G = (\Sigma, \mathcal{O}, X_0, \delta)$ such that $M_G \sim G$.

First, we show the construction of the automaton M_G.

DEFINITION 4.9
(The construction of M_G)
Let $G = (\Sigma, \mathcal{B}, \tau_0, \alpha)$ be an **L**-grammar. We construct an **L**-automaton $M_G = (\Sigma, \mathcal{O}, X_0, \delta)$ as follows:

- $\mathcal{O} = ST(\{\tau_0\} \cup \bigcup_{\sigma\in\Sigma} \alpha[\sigma])$
- $X_0 = \tau_0$
- δ is defined as follows:
  - for every $\sigma \in \Sigma$ and $\tau \in \mathcal{O}$ such that $\tau \in \alpha[\sigma]$: shift$[\sigma, \tau]$;
  - for every $\tau \in \mathcal{O}$: hypothesis-shift$[\tau]$;
  - for every $\tau_1, \tau_2, \tau \in \mathcal{O}$ such that either $\tau_2 = (\tau_1 \to \tau)$ or $\tau_1 = (\tau \leftarrow \tau_2)$: reduce$[\tau_1, \tau_2, \tau]$;
  - for every $\tau_1, \tau_2, \tau \in \mathcal{O}$ such that $\tau_1 = (\tau_2 \to \tau)$: left-release$[\tau_1, \tau_2, \tau]$;
  - for every $\tau_1, \tau_2, \tau \in \mathcal{O}$ such that $\tau_1 = (\tau \leftarrow \tau_2)$: right-release$[\tau_1, \tau_2, \tau]$.

EXAMPLE 4.10
We construct the automaton M_{G_2} from the grammar G_2, described in example 2.8. $M_{G_2} = (\Sigma, \mathcal{O}, X_0, \delta)$, where:

- $\Sigma = \{a, b\}$,
- $\mathcal{O} = \{A, B, C, (B \to A), (B \to C), ((B \to A) \to (B \to C)), ((B \to C) \leftarrow B), (C \to A)\}$
  For a shorter notation, we denote each object in $\mathcal{O}$ by some letter X_i:
  $X_0 \equiv A, X_1 \equiv B, X_2 \equiv ((B \to C) \leftarrow B), X_3 \equiv (B \to C), X_4 \equiv C, X_5 \equiv (C \to A)$, $X_6 \equiv (B \to A), X_7 \equiv ((B \to A) \to (B \to C))$
- δ contains the following transitions:
  - shift$[a, X_1]$, shift$[a, X_7]$, shift$[b, X_2]$, shift$[b, X_5]$;
  - hypothesis-shift$[X_0]$, hypothesis-shift$[X_1]$, hypothesis-shift$[X_2]$, hypothesis-shift$[X_3]$, hypothesis-shift$[X_4]$, hypothesis-shift$[X_5]$, hypothesis-shift$[X_6]$, hypothesis-shift$[X_7]$;
  - reduce$[X_2, X_1, X_3]$, reduce$[X_1, X_3, X_4]$, reduce$[X_4, X_5, X_0]$, reduce$[X_6, X_7, X_3]$, reduce$[X_1, X_6, X_0]$;
  - left-release$[X_3, X_1, X_4]$, left-release$[X_5, X_4, X_0]$, left-release$[X_6, X_1, X_0]$, left-release$[X_7, X_6, X_3]$;
  - right-release$[X_2, X_1, X_3]$.

Recall the automaton M_1 from example 3.3 and compare it to our resulting automaton M_{G_2} (after renaming its objects in $\mathcal{O}$ to X_i $(0 \leq i \leq 7)$). Note that M_1 has one redundant object X_8 which does not exist in M_{G_2}.

In order to prove theorem 4.8 we have to prove that M_G is an **L**-automaton and that M_G and G are tightly related. Let $R_{\approx}$ be the equivalence relation, as specified in definition 3.1 and let Π be the set of equivalence classes of $R_{\approx}$. First, we prove an auxiliary proposition stating that the only pairs in $R_{\approx}$ are trivial pairs of the same objects.

PROPOSITION 4.11
$R_{\approx} = I_{\mathcal{O}}$

PROOF. We iteratively build the relation $R_{\approx}$. First, $R_{\approx} = \{(X, X) \mid X \in \mathcal{O}\}$, i.e., $R_{\approx} = I_{\mathcal{O}}$. Let $left - release\,[X_1, Y_1, Z_1]$ and $left - release\,[X_2, Y_2, Z_2]$. By construction of M_G, $X_1 = (\tau_1 \to \tau_2)$, where $Y_1 = \tau_1, Z_1 = \tau_2$, whereas $X_2 = (\tau_1' \to \tau_2')$, where $Y_2 = \tau_1', Z_2 = \tau_2'$. Thus, $X_1 = X_2$ iff $Y_1 = Y_2$ and $Z_1 = Z_2$. Consequently, $left - release$ transitions do not add new pairs to $R_{\approx}$. Similarly, $right - release$ transitions do not add new pairs to $R_{\approx}$. Finally, since $R_{\approx} = I_{\mathcal{O}}$, then the transitive closure of $R_{\approx}$ is $R_{\approx}$ itself.

Thus, we obtained that $R_{\approx} = I_{\mathcal{O}}$. ∎

PROPOSITION 4.12
Let $M_G = (\Sigma, \mathcal{O}, X_0, \delta)$ be an automaton constructed from an **L**-grammar $G = (\Sigma, \mathcal{B}, \tau_0, \alpha)$ by definition 4.9. Then M_G satisfies the constraints for **L**-automata.

PROOF. We verify the **L**-automaton constraints.

1. By the definition of M_G, for every $\tau \in \mathcal{O}$, $hypothesis - shift\,[\tau]$. In particular, for every $[X] \in \Pi$ there is $X_1 \in [X]$ such that $hypothesis - shift\,[X_1]$;

2. Let $left-release\,[X_1, Y_1, Z_1]$ and $right-release\,[X_2, Y_2, Z_2]$. By construction of M_G, $X_1 = (\tau_1 \to \tau_2)$, where $Y_1 = \tau_1, Z_1 = \tau_2$, whereas $X_2 = (\tau_2' \leftarrow \tau_1')$, where $Y_2 = \tau_1', Z_2 = \tau_2'$. Thus, $X_1 \neq X_2$. Proposition 4.11 implies that also $[X_1] \neq [X_2]$.
3. Constraint 3 follows trivially from proposition 4.11.
4. By the construction of M_G, $reduce\,[\tau_1, \tau_2, \tau]$ iff $\tau_1, \tau_2, \tau \in \mathcal{O}$ and either $\tau_2 = (\tau_1 \to \tau)$ or $\tau_1 = (\tau \leftarrow \tau_2)$ iff either $left-release\,[\tau_2, \tau_1, \tau]$ or $right-release\,[\tau_1, \tau_2, \tau]$.
5. We prove that for every $([\tau'], [\tau'']) \in D_{M_G}$, $\#(\tau') > \#(\tau'')$ ($D_{M_G} \cap I_\Pi = \emptyset$ follows). This claim is well-defined, since equivalence classes are singletons by proposition 4.11.
   Let $([\tau'], [\tau'']) \in D_{M_G}$. Then there is the maximal sequence $\tau_1, \ldots, \tau_n \in \mathcal{O}$ such that $([\tau'], [\tau_1]) \in D_{M_G}$, $([\tau_1], [\tau_2]) \in D_{M_G}, \ldots, ([\tau_n], [\tau'']) \in D_{M_G}$. We prove the claim by induction on n. Let

$$R = \{(X, Y, Z) \mid left-release\,[X, Y, Z] \text{ or } right-release\,[X, Y, Z]\}$$

   - $n = 0$: There is $\tau \in \mathcal{O}$ such that either $(\tau', \tau'', \tau) \in R$ or $(\tau', \tau, \tau'') \in R$. By the definition of M_G, either $\tau' = (\tau'' \to \tau)$ or $\tau' = (\tau'' \leftarrow \tau)$ or $\tau' = (\tau \to \tau'')$ or $\tau' = (\tau \leftarrow \tau'')$. Therefore, $\#(\tau') = \#(\tau) + \#(\tau'')$, and thus $\#(\tau') > \#(\tau'')$.
   - $([\tau'], [\tau'']) \in D_{M_G}$, where there is $\rho \in \mathcal{O}$, s.t. $([\tau'], [\rho]) \in D_{M_G}$ and $([\rho], [\tau'']) \in D_{M_G}$. By the induction hypothesis, $\#(\tau') > \#(\rho)$ and $\#(\rho) > \#(\tau'')$. $\#(\tau') > \#(\tau'')$ follows.

∎

In order to prove that $M_G \sim G$, we need to find a mapping T from $\mathcal{O}$ into $\mathcal{C}$ satisfying all the conditions as required by definition 4.4. Let T be the identity function $I : \mathcal{O} \longrightarrow \mathcal{O}$ (recall that $\mathcal{O} = ST(\{\tau_0\} \cup \bigcup_{\sigma \in \Sigma} \alpha[\sigma])$).
We have to prove that for every marked normal witness derivation Δ in G there is a run π of M_G such that $\pi \sim_T \Delta$, and in the opposite direction, for every accepting run π of M_G there is a marked **L**-derivation Δ such that $\pi \sim_T \Delta$. We use the following propositions to prove the two directions.

PROPOSITION 4.13
Let Δ be a subtree of a marked normal witness derivation in G:

$$\Delta = \langle (v_1, \text{children}(v_1), \Gamma_1 \rhd \tau_1), \ldots, (v_k, \text{children}(v_k), \Gamma_k \rhd \tau_k) \rangle$$

Then, there is a run $\pi = c_{ini} \vdash^k c_k$ of M_G such that $\pi \sim_T \Delta$.

PROOF. First, we show that for every category τ appearing in Δ, $\tau \in \mathcal{O}$.
Let Δ' be a marked normal witness derivation containing Δ, and let $\Gamma \rhd \tau_0$ be the sequent that is proved by Δ', where $\Gamma \in \alpha[w]$ for some $w \in \Sigma^+$. Then $\tau \in \Delta'$, and by the subformula property τ is a subcategory of τ_0 or of some τ_i in Γ. Since $\Gamma \in \alpha[w]$, then $\tau \in ST(\{\tau_0\} \cup \bigcup_{\sigma \in \Sigma} \alpha[\sigma]) = \mathcal{O}$.

We proceed by induction on k - the length of Δ.

$k = 1$: Δ is the axiom, either $\tau_1 \rhd \tau_1$ (where $\tau_1 \in \alpha[\sigma]$ for $\sigma = \text{children}(v_1)$) or $[\tau_1] \rhd \tau_1$. Note, that $\tau_1 \in \mathcal{O}$.

- In the former case, by the definition of M_G, δ contains the transition $shift\,[\sigma, \tau_1]$. We choose π to be the above $shift$ transition: $\pi = \langle \sigma, \epsilon \rangle \vdash^{shift} \langle \epsilon, (\tau_1, \star) \rangle$. $\tau_1 = T(\tau_1)$ implies that $\pi \sim_T \Delta$.

- In the latter case, we choose π to be the transition $hypothesis - shift\,[\tau_1]$ (recall that $hypothesis-shift$ transitions are defined for every $\tau \in \mathcal{O}$, in particular for τ_1): $\pi = \langle \epsilon, \epsilon \rangle \vdash^{hypothesis-shift} \langle \epsilon, (\tau_1, \tau_1) \rangle$. $\tau_1 = T(\tau_1)$ implies that $\pi \sim_T \Delta$.

Suppose the proposition holds for every Δ of length $\leq k$ ($k > 1$), and let Δ be a subtree of a marked normal witness derivation tree of length $k+1$:

$$\Delta = \langle (v_1, \text{children}(v_1), \Gamma_1 \rhd \tau_1), \ldots, (v_{k+1}, \text{children}(v_{k+1}), \Gamma_{k+1} \rhd \tau_{k+1}) \rangle$$

There are three possibilities for the last inference rule applied in Δ:

- <u>arrow elimination:</u>
  Δ is obtained from two subtrees Δ_1 and Δ_2 by an arrow elimination rule application:

$$\Delta: \quad \frac{\Delta_1: \ \dfrac{\vdots}{\Gamma_i \rhd \tau_i} \qquad \Delta_2: \ \dfrac{\vdots}{\Gamma_k \rhd \tau_k}}{\Gamma_{k+1} \rhd \tau_{k+1}}$$

  (Recall that by the linearization order, the root of Δ is preceded by the root of Δ_2; therefore, the roots of Δ_2 is indexed by k). By the induction hypothesis there is a run π_1 of M_G such that $\pi_1 \sim_T \Delta_1$:
  $\pi_1 = \langle u, \epsilon \rangle \vdash^i \langle \epsilon, \mu \cdot (\tau_i, \beta_i) \rangle$, and there is a run π_2 of M_G such that $\pi_2 \sim_T \Delta_2$:
  $\pi_2 = \langle v, \epsilon \rangle \vdash^{k-i} \langle \epsilon, \nu \cdot (\tau_k, \beta_k) \rangle$. By proposition 4.3, π_1 and π_2 are balanced runs, i.e. $\mu = \nu = \epsilon$. In order to construct one run from π_1 and π_2, we "wrap" π_1 and π_2 as follows.
  – For π_1, we add the input word v after the given input u:

$$\pi_1' = \langle uv, \epsilon \rangle \vdash^i \langle v, (\tau_i, \beta_i) \rangle$$

    By proposition 3.7, the resulting run π_1' has the same moves as π_1, implying that $\pi_1' \sim_T \Delta_1$.
  – For π_2, we add (τ_i, β_i) at the bottom of HS before the run starts:

$$\pi_2' = \langle v, (\tau_i, \beta_i) \rangle \vdash^{k-i} \langle \epsilon, (\tau_i, \beta_i)(\tau_k, \beta_k) \rangle$$

    By proposition 3.7, the resulting run π_2' has the same moves as π_2, implying that $\pi_2' \sim_T \Delta_2$.
  Since the last configuration of π_1' is exactly the first configuration of π_2', π_2' may be performed after π_1'. Furthermore, since $\tau_i, \tau_k, \tau_{k+1} \in \mathcal{O}$ and either $\tau_k = (\tau_i \rightarrow \tau_{k+1})$ or $\tau_i = (\tau_{k+1} \leftarrow \tau_k)$ then, by the definition of M_G, reduce$[\tau_i, \tau_k, \tau_{k+1}]$. We construct π by first running π_1', then running π_2' and finally performing the above reduce transition:

$$\begin{aligned} \pi = \quad & \langle uv, \epsilon \rangle \vdash^i \langle v, (\tau_i, \beta_i) \rangle \vdash^{k-i} \langle \epsilon, (\tau_i, \beta_i)(\tau_k, \beta_k) \rangle \vdash^{reduce} \\ & \langle \epsilon, (\tau_{k+1}, \beta_{k+1}) \rangle \end{aligned}$$

  Thus, the first k steps of π correspond to the first k inference rules of Δ, as required by definition 4.1; $\tau_{k+1} = T(\tau_{k+1})$ implies that also the last step of π corresponds to the last inference rule of Δ. Thus, $\pi \sim_T \Delta$.

- arrow introduction $(\rightarrow I)$:
  Δ is obtained from the subtree Δ_1 by an arrow introduction $(\rightarrow I)$ rule application:

$$\Delta: \quad \frac{\Delta_1: \quad \begin{array}{c}\vdots\\ \overline{[\tau]\Gamma \rhd \tau_k}\end{array}}{\Gamma_{k+1} \rhd \tau_{k+1}} \ (\rightarrow I) \quad (\Gamma \neq \epsilon)$$

  where $\tau_{k+1} = (\tau \rightarrow \tau_k)$ and $\Gamma_{k+1} = \Gamma$. By the induction hypothesis there is a run π_1 of M_G such that $\pi_1 \sim_T \Delta_1$: $\pi_1 = \langle u, \epsilon\rangle \vdash^i \langle \epsilon, \mu \cdot (\tau_k, \beta)\rangle$. By proposition 4.3 π_1 is a balanced run, i.e. $\mu = \epsilon$; moreover, the same proposition implies that
  – either $[\tau]\Gamma = [\tau\Gamma^l]\hat{\Gamma}'[\Gamma^r]$ $(\Gamma' \neq \epsilon)$ and $\beta = \beta^l \star \beta^r$, where $\tau\Gamma^l = T(\beta^l) = \beta^l$ and $\Gamma^r = T(\beta^r) = \beta^r$,
  – or $[\tau]\Gamma = [\tau\Gamma']$ and $\tau\Gamma' = T(\beta) = \beta$.
  Consequently, $\beta = \tau\beta'$. Furthermore, since $\tau_{k+1} = (\tau \rightarrow \tau_k)$ and $\tau_{k+1}, \tau, \tau_k \in \mathcal{O}$ then, by the definition of M_G, left-release$[\tau_{k+1}, \tau, \tau_k]$. We construct π by first running π_1 and then performing the above left-release transition:

$$\pi = \langle u, \epsilon\rangle \vdash^k \langle \epsilon, (\tau_k, \beta_k)\rangle \vdash^{left-release} \langle \epsilon, (\tau_{k+1}, \beta_{k+1})\rangle$$

  Thus, the first k steps of π correspond to the first k inference rules of Δ, as required by definition 4.1; $\tau_{k+1} = T(\tau_{k+1})$ implies that also the last step of π corresponds to the last inference rule of Δ. Thus, $\pi \sim_T \Delta$.
- arrow introduction $(\leftarrow I)$:
  This case is symmetric to the previous case $(\rightarrow I)$.

∎

PROPOSITION 4.14
Let $\pi = c_0 \vdash^k c_k$ be a subrun of an accepting run of M_G, such that π is balanced. Then, there is the marked **L**-derivation Δ such that $\pi \sim_T \Delta$.

PROOF. First, recall that $\tau = T(\tau)$ for every $\tau \in \mathcal{O}$, since $T : \mathcal{O} \longrightarrow \mathcal{O}$ is the identity function.

We prove the proposition by induction on k, the length of π.

$k = 1$: since π is a balanced run, it adds one element onto HS. Therefore, the single-step balanced-run is a $shift$ or a $hypothesis - shift$ transition.

- In the former case, let the move be $shift[\sigma, \tau]$. By the definition of M_G, $\tau \in \alpha[\sigma]$. We choose Δ to be the axiom $\tau \rhd \tau$ with σ at the extra level child node: $\Delta = \langle v, \sigma, \tau \rhd \tau\rangle$. $\tau = T(\tau)$ implies that $\pi \sim_T \Delta$.
- In the latter case, let the move be $hypothesis - shift[\tau]$. We choose Δ to be the axiom $[\tau] \rhd \tau$. $\tau = T(\tau)$ implies that $\pi \sim_T \Delta$.

Suppose the proposition holds for balanced runs of length $\leq k$ $(k > 1)$, and let $\pi = c_0 \vdash^{k+1} c_{k+1}$ be a balanced run of length $k+1$: $\pi = \langle w, \mu\rangle \vdash^{k+1} \langle w_{k+1}, \mu \cdot (\tau, \beta)\rangle$. There are five possibilities for the last move of π:

- shift and hypothesis-shift: Both are impossible for a balanced runs of length ≥ 2 by proposition 3.11.

- reduce: The move $c_k \vdash c_{k+1}$ is $\langle w_k, \mu \cdot (\tau_1, \beta_1)(\tau_2, \beta_2)\rangle \vdash^{\text{reduce}} \langle w_k, \mu \cdot (\tau, \beta)\rangle$. Consider the prefix of π of length k: $\langle w, \mu\rangle \vdash^k \langle w_k, \mu \cdot (\tau_1, \beta_1)(\tau_2, \beta_2)\rangle$. By proposition 3.10, it may be partitioned into two balanced runs π_1 and π_2, where π_1 adds (τ_1, β_1) into HS and π_2 adds (τ_2, β_2) into HS:

$$\pi_1 = \langle w, \mu\rangle \vdash^i \langle w_i, \mu \cdot (\tau_1, \beta_1)\rangle$$
$$\pi_2 = \langle w_i, \mu \cdot (\tau_1, \beta_1)\rangle \vdash^{k-i} \langle w_k, \mu \cdot (\tau_1, \beta_1)(\tau_2, \beta_2)\rangle$$

  By the induction hypothesis there are marked **L**-derivations Δ_1 and Δ_2, such that $\pi_1 \sim_T \Delta_1$ and $\pi_2 \sim_T \Delta_2$. Thus, $\text{root}(\Delta_1) = \Gamma_1 \rhd \tau_1$ and $\text{root}(\Delta_2) = \Gamma_2 \rhd \tau_2$, where Γ_1 and Γ_2 are some labeled sequences of categories. Since π is a subrun of an accepting run, then β satisfies the conditions of proposition 3.6, implying that
  – $\beta = \gamma_1 \star \gamma_2$, where $\gamma_1, \gamma_2 \in \mathcal{O}^\star$
  or
  – $\beta \in \mathcal{O}^+$
  Since β_1, β_2 also satisfy the above conditions, we have the following possibilities for the structure of β_1, β_2.
  1. $\beta_1 = \beta_1^l \star \beta_1^r$ ($\beta_1^l, \beta_1^r \in \mathcal{O}^\star$) and $\beta_2 \in \mathcal{O}^+$.
     Then, by proposition 4.3, $\Gamma_1 = [\Gamma_1^l]\hat{\Gamma}_1'[\Gamma_1^r]$ and $\Gamma_2 = [\Gamma_2']$, where $\Gamma_1^l = T(\beta_1^l)$, $\Gamma_1^r = T(\beta_1^r)$, and $\Gamma_2' = T(\beta_2)$. Thus, $\Gamma_1\Gamma_2 = [\Gamma_1^l]\hat{\Gamma}_1'[\Gamma_1^r\Gamma_2']$.
  2. $\beta_1 = \beta_1^l \star \beta_1^r$ ($\beta_1^l, \beta_1^r \in \mathcal{O}^\star$) and $\beta_2 \in \mathcal{O}^+$.
     Then, symmetrically, $\Gamma_1\Gamma_2 = [\Gamma_1'\Gamma_2^l]\hat{\Gamma}_2'[\Gamma_2^r]$.
  3. $\beta_1 = \beta_1^l \star$ ($\beta_1^l \in \mathcal{O}^\star$) and $\beta_2 = \star\beta_2^r$ ($\beta_2^r \in \mathcal{O}^\star$).
     Then, by proposition 4.3, $\Gamma_1 = [\Gamma_1^l]\hat{\Gamma}_1'$ and $\Gamma_2 = \hat{\Gamma}_2'[\Gamma_2^r]$, where $\Gamma_1^l = T(\beta_1^l)$ and $\Gamma_2^r = T(\beta_2^r)$. Thus, $\Gamma_1\Gamma_2 = [\Gamma_1^l]\hat{\Gamma}_1'\hat{\Gamma}_2'[\Gamma_2^r]$.
  4. Finally, $\beta_1, \beta_2 \in \mathcal{O}^+$.
     Then, by proposition 4.3, $\Gamma_1 = [\Gamma_1']$ and $\Gamma_2 = [\Gamma_2']$, where $\Gamma_1' = T(\beta_1)$ and $\Gamma_2' = T(\beta_2)$. Thus, $\Gamma_1\Gamma_2 = [\Gamma_1'\Gamma_2']$.

  To sum up, the labeled sequence $\Gamma_1\Gamma_2$ satisfies the side condition for applying the arrow elimination rule on the reduction statements $\Gamma_1 \rhd \tau_1$ and $\Gamma_2 \rhd \tau_2$. Furthermore, by the definition of M_G, either $\tau_1 = (\tau \leftarrow \tau_2)$ or $\tau_2 = (\tau_1 \rightarrow \tau)$; thus, an arrow elimination rule is applicable on $\Gamma_1 \rhd \tau_1$ and $\Gamma_2 \rhd \tau_2$, obtaining the new marked derivation tree Δ:

$$\Delta: \quad \frac{\Delta_1: \ \dfrac{\vdots}{\Gamma_1 \rhd \tau_1} \qquad \Delta_2: \ \dfrac{\vdots}{\Gamma_2 \rhd \tau_2}}{\Gamma_1\Gamma_2 \rhd \tau}$$

  where
  – the prefix of Δ, Δ_1, satisfies $\pi_1 \sim_T \Delta_1$;
  – the next part of Δ, Δ_2, satisfies $\pi_2 \sim_T \Delta_2$;
  – the last inference rule of Δ is T-related to the last move of π, since $\tau = T(\tau)$.
  Therefore, $\pi \sim_T \Delta$.
- left-release: The move $c_k \vdash c_{k+1}$ is $\langle w_k, \mu \cdot (\tau_1, \tau_2\beta)\rangle \vdash^{\text{left-release}} \langle w_k, \mu \cdot (\tau, \beta)\rangle$. Consider the prefix π_1 of π of length k: $\pi_1 = \langle w, \mu\rangle \vdash^k \langle w_k, \mu \cdot (\tau_1, \tau_2\beta)\rangle$. Since π is a balanced run, then $c_i = \langle w_i, \mu\mu_i\rangle$ for every $1 \le i \le k$, implying that π_1 is a balanced run too. By the induction hypothesis there is a marked tree Δ_1 such that $\pi_1 \sim_T \Delta_1$. $\text{root}(\Delta_1) = \Gamma_1 \rhd \tau_1$ for some Γ_1; then, by proposition 4.3,

– either $\Gamma_1 = [\Gamma_1^l]\hat{\Gamma}_1'[\Gamma_1^r]$ ($\Gamma_1' \neq \epsilon$) and $\tau_2\beta = \beta_l \star \beta_r$, where $\Gamma_1^l = T(\tau_2\beta_l) = \tau_2\beta_l$, and $\Gamma_1^r = T(\beta_r) = \beta_r$
– or $\Gamma_1 = [\Gamma_1']$ and $\Gamma_1' = T(\tau_2\beta) = \tau_2\beta$.

Consequently, $\Gamma_1 = [\tau_2]\Gamma$. Furthermore, by the definition of M_G, $\tau = (\tau_2 \to \tau_1)$; thus, an arrow introduction rule is appliable to $\Gamma_1 \rhd \tau_1$, obtaining the new marked derivation tree Δ:

$$\Delta: \quad \frac{\Delta_1: \quad \dfrac{\vdots}{[\tau_2]\Gamma \rhd \tau_1}}{\Gamma \rhd \tau}\ (\to I)$$

where
– the prefix of Δ, Δ_1, satisfies $\pi_1 \sim_T \Delta_1$;
– the last inference rule of Δ is T-related to the last move of π, since $\tau = T(\tau)$.

Therefore, $\pi \sim_T \Delta$.

- right-release: This case is symmetric to left-release.

∎

PROOF. **of theorem 4.8**
We construct M_G as in definition 4.9. Proposition 4.12 implies that M_G is an **L**-automaton. Let T be the identity function $I : \mathcal{O} \longrightarrow \mathcal{O}$. Then, $\tau_0 = X_0 = I(X_0)$ by the definition of M_G. Let Δ be a marked normal witness derivation in G. By proposition 4.13, there is a run $\pi = c_{ini} \vdash^k c_k$ of M_G such that $\pi \sim_T \Delta$.
Let $\pi = c_{ini} \vdash^n c_{acc}$ be an accepting run of M_G. Naturally, π is a balanced run; this implies, by proposition 4.14, that there is a marked **L**-derivation Δ such that $\pi \sim_T \Delta$. ∎

4.4 Constructing an **L***-grammar from an* **L***-automaton*

In this section we show how to construct an **L**-grammar tightly related to a given **L**-automaton. The remark about (non-)minimality applies here too.

THEOREM 4.15
For every **L**-automaton $M = (\Sigma, \mathcal{O}, X_0, \delta)$ there is an **L**-grammar $G_M = (\Sigma, \mathcal{B}, \tau_0, \alpha)$ such that $M \sim G_M$.

We start by describing the *Category-Resolution Algorithm*, which is used for the construction of G_M.

4.4.1 Category-Resolution Algorithm

The algorithm is presented in Figure 10. It gets an **L**-automaton M and returns a set of basic categories $\mathcal{B}$ and a mapping $T : \mathcal{O} \to \mathcal{C}$ assigning a category over $\mathcal{B}$ to every object in $\mathcal{O}$.

The algorithm works in iterations. Initially, the relation R_{eq} is calculated (actually, R_{eq} is the equivalence relation $R_\approx$, which will be further proved). Note that generally, the equivalence relation $R_\approx$ is defined as a part of an **L**-automaton (see definition 3.1) and need not be calculated by the algorithm. However, here we present a more

Category-Resolution Algorithm

input: an **L**-automaton $M = (\Sigma, \mathcal{O}, X_0, \delta)$;
output: a non-empty set of basic categories $\mathcal{B}$;
a function $T : \mathcal{O} \to \mathcal{C}$ such that for every $X \in \mathcal{O}$:

- $T(X) \in \mathcal{B}$ iff for all $Y, Z \in \mathcal{O}$
  not left-release$[X, Y, Z]$ and not right-release$[X, Y, Z]$;
- for every $Y, Z \in \mathcal{O}$:
  $T(X) = (T(Y) \to T(Z))$ iff left-release$[X, Y, Z]$;
  $T(X) = (T(Z) \leftarrow T(Y))$ iff right-release$[X, Y, Z]$;
- for every $Y \in \mathcal{O}$: $[X] = [Y]$ iff $T(X) = T(Y)$.

1. $S_{uniq} = \emptyset$
   $R_l = \{(X, Y, Z) \mid \text{left-release}\,[X, Y, Z]\}$
   $R_r = \{(X, Y, Z) \mid \text{right-release}\,[X, Y, Z]\}$
   $R_{eq} = \{(X_1, X_2) \mid$ there are $Y, Z \in \mathcal{O}$
   such that either $(X_1, Y, Z), (X_2, Y, Z) \in R_l$
   or $(X_1, Y, Z), (X_2, Y, Z) \in R_r$
   or $(Y, X_1, Z), (Y, X_2, Z) \in R_l \cup R_r$
   or $(Y, Z, X_1), (Y, Z, X_2) \in R_l \cup R_r\} \cup \{(X, X) \mid X \in \mathcal{O}\}$
2. while there is $X_1 \in (\mathcal{O} \backslash S_{uniq})$ such that for every $X_2 \in S_{uniq}$
   $(X_1, X_2) \notin R_{eq}$: $S_{uniq} = S_{uniq} \cup \{X_1\}$
3. $\mathcal{B} = \{\tau_X \mid X \in S_{uniq}$ and for every $Y, Z \in S_{uniq}$ $(X, Y, Z) \notin R_l \cup R_r\}$
4. for every $X \in S_{uniq}$ such that $\tau_X \in \mathcal{B}$: $T(X) = \tau_X$
5. while there is a triple $(X, Y, Z) \in R_l \cup R_r$ such that $X \in S_{uniq}$ and $T(X)$ is undefined, whereas $T(Y)$ and $T(Z)$ are already defined, extend the mapping T to X:
$$T(X) = \left\{ \begin{array}{ll} (T(Y) \to T(Z)) & \text{if } (X, Y, Z) \in R_l \\ (T(Z) \leftarrow T(Y)) & \text{if } (X, Y, Z) \in R_r \end{array} \right\}$$
6. while there is a pair $(X_1, X_2) \in R_{eq}$ such that $T(X_1)$ is undefined whereas $T(X_2)$ is already defined, extend the mapping T to X_1:
$$T(X_1) = T(X_2)$$
7. return $\mathcal{B}$ and T

FIG. 10. The Category-Resolution Algorithm

efficient calculation of $R_{\approx}$, based on the fact that the constraints of the **L**-automaton are satisfied.

Then, the algorithm calculates the set S_{uniq}, which is the subset of $\mathcal{O}$ containing one representative from each equivalence class of R_{eq}.

Next, the algorithm iteratively calculates categories for objects in S_{uniq}. First, T includes only mappings to the basic categories. In every iteration the mapping T is extended, when complex categories are built from simpler ones.

When every object in S_{uniq} is mapped to some category over $\mathcal{B}$, the algorithm proceeds to the final stage. In the final stage, the algorithm iteratively calculates categories for the rest of objects in $\mathcal{O}$. For each object it finds in S_{uniq} a representative of its equivalence class and assigns a representative's category to the object.

When every object in $\mathcal{O}$ is mapped to some category over $\mathcal{B}$, the algorithm terminates returning the set of basic categories $\mathcal{B}$ and the mapping T.

Before proving the correctness of the algorithm we demonstrate how it runs.

EXAMPLE 4.16
We run the Category-Resolution algorithm for the **L**-automaton M_1 from example 3.3.

1. $S_{uniq} = \emptyset$

$$\begin{aligned} R_l = \quad & \{(X_3, X_1, X_4), (X_3, X_8, X_4), (X_5, X_4, X_0), (X_6, X_1, X_0), (X_6, X_8, X_0), \\ & (X_7, X_6, X_3)\} \\ R_r = \quad & \{(X_2, X_1, X_3), (X_2, X_8, X_3)\} \\ R_{eq} = \quad & \{(X_0, X_0), (X_1, X_1), (X_2, X_2), (X_3, X_3), (X_4, X_4), (X_5, X_5), \\ & (X_6, X_6), (X_7, X_7), (X_8, X_8), (X_1, X_8), (X_8, X_1)\} \end{aligned}$$

2. $S_{uniq} = \{X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7\}$
3. $\mathcal{B} = \{\tau_{X_0}, \tau_{X_1}, \tau_{X_4}\}$
4. $T(X_0) = \tau_{X_0},\ T(X_1) = \tau_{X_1},\ T(X_4) = \tau_{X_4}$;
5. • found triple (X_3, X_1, X_4); $T(X_3) = (\tau_{X_1} \to \tau_{X_4})$;
   • found triple (X_5, X_4, X_0); $T(X_5) = (\tau_{X_4} \to \tau_{X_0})$;
   • found triple (X_6, X_1, X_0); $T(X_6) = (\tau_{X_1} \to \tau_{X_0})$;
   • found triple (X_7, X_6, X_3); $T(X_7) = ((\tau_{X_1} \to \tau_{X_0}) \to (\tau_{X_1} \to \tau_{X_4}))$;
   • found triple (X_2, X_1, X_3); $T(X_2) = ((\tau_{X_1} \to \tau_{X_4}) \leftarrow \tau_{X_1})$;
6. found pair (X_8, X_1); $T(X_8) = T(X_1) = \tau_{X_1}$
7. return $\mathcal{B}$ and T.

The following propositions prove the correctness of the algorithm.
Correctness of step 1: the relation R_{eq} is transitive and is equal to the equivalence relation $R_{\approx}$.

PROPOSITION 4.17
$R_{eq} \subseteq R_{\approx}$

PROOF. Let $(X_1, X_2) \in R_{eq}$. If $X = Y$ then by the definition of $R_{\approx}$, $(X_1, X_2) \in R_{\approx}$. Otherwise, there are $Y, Z \in \mathcal{O}$ such that one of the following holds:

- $(X_1, Y, Z), (X_2, Y, Z) \in R_l$. Then by the definition of R_l, $left - release\,[X_1, Y, Z]$ and $left - release\,[X_2, Y, Z]$. By the definition of $R_{\approx}$, $(Y, Y) \in R_{\approx}$ and $(Z, Z) \in R_{\approx}$, which implies that $(X_1, X_2) \in R_{\approx}$.

- $(X_1, Y, Z), (X_2, Y, Z) \in R_r$: similar.
- $(Y, X_1, Z), (Y, X_2, Z) \in R_l \cup R_r$. Since $(Y, Y) \in R_{\approx}$, then constraint 2 of M implies that either $left - release\,[Y, X_1, Z]$, $left - release\,[Y, X_2, Z]$ or $right - release\,[Y, X_1, Z]$, $right - release\,[Y, X_2, Z]$. By the definition of $R_{\approx}$, $(Y, Y) \in R_{\approx}$ implies that $(X_1, X_2) \in R_{\approx}$.
- $(Y, Z, X_1), (Y, Z, X_2) \in R_l \cup R_r$: similar.

∎

PROPOSITION 4.18
R_{eq} is transitive.

PROOF. Let $(X_1, X_2) \in R_{eq}$ and $(X_2, X_3) \in R_{eq}$.
$(X_1, X_2) \in R_{eq}$ implies that there are $Y, Z \in \mathcal{O}$ such that either $(X_1, Y, Z), (X_2, Y, Z) \in R_l$ or $(X_1, Y, Z), (X_2, Y, Z) \in R_r$ or $(Y, X_1, Z), (Y, X_2, Z) \in R_l \cup R_r$ or $(Y, Z, X_1), (Y, Z, X_2) \in R_l \cup R_r$. Since $R_{eq} \subseteq R_{\approx}$, then $(X_2, X_3) \in R_{\approx}$. This implies, by constraint 3 of M, that either $(X_3, Y, Z) \in R_l$ or $(X_3, Y, Z) \in R_r$ or $(Y, X_3, Z) \in R_l \cup R_r$ or $(Y, Z, X_3) \in R_l \cup R_r$. Therefore, $(X_1, X_3) \in R_{eq}$. ∎

PROPOSITION 4.19
$R_{\approx} \subseteq R_{eq}$

PROOF. Let $(X_1, X_2) \in R_{\approx}$. By the definition of $R_{\approx}$, there are three possibilities:

1. $X_1 = X_2$: then, by the definition of R_{eq}, $(X_1, X_2) \in R_{eq}$.
2. There are $Y_1, Y_2, Z_1, Z_2 \in \mathcal{O}$ such that one of the following holds:
   - $left{-}release\,[X_1, Y_1, Z_1], left{-}release\,[X_2, Y_2, Z_2]$ or $right{-}release\,[X_1, Y_1, Z_1]$, $right{-}release\,[X_2, Y_2, Z_2]$, where $[Y_1] = [Y_2]$ and $[Z_1] = [Z_2]$. Then, constraint 3 implies that $left - release\,[X_2, Y_1, Z_1]$ or $right - release\,[X_2, Y_1, Z_1]$. In other words, $(X_1, Y_1, Z_1), (X_2, Y_1, Z_1) \in R_l$ or $(X_1, Y_1, Z_1), (X_2, Y_1, Z_1) \in R_r$. Therefore, by the definition of R_{eq}, $(X_1, X_2) \in R_{eq}$.
   - $left{-}release\,[Y_1, X_1, Z_1], left{-}release\,[Y_2, X_2, Z_2]$ or $right{-}release\,[Y_1, X_1, Z_1]$, $right - release\,[Y_2, X_2, Z_2]$, where $[Y_1] = [Y_2]$. Then, constraint 3 implies that $left - release\,[Y_1, X_2, Z_2]$ or $right - release\,[Y_1, X_2, Z_2]$. In other words, $(Y_1, X_1, Z_1), (Y_1, X_2, Z_2) \in R_l$ or $(Y_1, X_1, Z_1), (Y_1, X_2, Z_2) \in R_r$. Therefore, by the definition of R_{eq}, $(X_1, X_2) \in R_{eq}$ (and $(Z_1, Z_2) \in R_{eq}$).
   - $left{-}release\,[Y_1, Z_1, X_1], left{-}release\,[Y_2, Z_2, X_2]$ or $right{-}release\,[Y_1, Z_1, X_1]$, $right - release\,[Y_2, Z_2, X_2]$, where $[Y_1] = [Y_2]$: similar.
3. There are $Y_1, \ldots Y_n \in \mathcal{O}$ such that $(X_1, Y_1) \in R_{\approx}, (Y_1, Y_2) \in R_{\approx}, \ldots, (Y_n, X_2) \in R_{\approx}$, where each pair from the above list is in the relation due to case 1 or 2 above. Then, as we proved above, $(X_1, Y_1) \in R_{eq}, (Y_1, Y_2) \in R_{eq} \ldots (Y_n, X_2) \in R_{eq}$. Transitivity of R_{eq} implies that $(X_1, X_2) \in R_{eq}$.

∎

Propositions 4.17 and 4.19 imply that $R_{eq} = R_{\approx}$, thus proving the correctness of step 1 of the algorithm.

Correctness of step 2: S_{uniq} contains exactly one representative of each equivalent class of $R_{\approx}$. In the following propositions we denote by S^0_{uniq} the initial content of S_{uniq} (which is $\emptyset$) and by S^i_{uniq} the content of S_{uniq} after the i-th iteration.

PROPOSITION 4.20
For every $i \geq 0$: for all $X, Y \in S^i_{uniq}$: $[X] \neq [Y]$.

PROOF. By induction on i.

- $i = 0$: $S^0_{uniq} = \emptyset$ and the proposition holds vacuously.
- Let $X, Y \in S^{i+1}_{uniq}$. If $X, Y \in S^i_{uniq}$ then by the induction hypothesis the condition holds. Otherwise, suppose without loss of generality that $X \notin S^i_{uniq}$. Then by the condition of step 2, for every $Z \in S^i_{uniq}$, $(X, Z) \notin R_{eq}$; in particular, $(X, Y) \notin R_{eq}$. Since $R_{eq} = R_{\approx}$ then $(X, Y) \notin R_{\approx}$, thus implying $[X] \neq [Y]$.

∎

PROPOSITION 4.21
The iteration in step 2 of the Category-Resolution Algorithm terminates.

PROOF. The iteration goes on as long as $X_1 \in (\mathcal{O} \backslash S_{uniq})$ (satisfying the conditions specified in the algorithm). In each iteration one object $X_1 \in \mathcal{O}$ is added to S_{uniq}. Since $\mathcal{O}$ is finite, the iteration stops after no more than $|\mathcal{O}|$ rounds. ∎

PROPOSITION 4.22
$S_{uniq} = \{X_1, \ldots, X_n\}$ $(n \geq 1)$ where $[X_i] \neq [X_j]$ for every $1 \leq i \neq j \leq n$ and $\bigcup_{1 \leq i \leq n} [X_i] = \Pi$.

PROOF. Let i be the number of rounds in step 2. By proposition 4.20 for all $X, Y \in S^i_{uniq}$: $[X] \neq [Y]$. Let $[Z] \in \Pi$ and suppose by way of contradiction that $[Z] \cap S^i_{uniq} = \emptyset$. Then for every $Z_1 \in [Z]$, for every $X \in S^i_{uniq}$: $(X, Z) \notin R_{eq}$, and thus the iteration may proceed with Z, which contradicts the fact that i is the last round. Finally, since $\mathcal{O} \neq \emptyset$ (because it contains at least the distinguished object X_0), then also $S_{uniq} \neq \emptyset$ (because it contains at least the equivalence class $[X_0]$). ∎

Correctness of step 3: the set of basic categories $\mathcal{B}$ is not empty.

PROPOSITION 4.23
$\mathcal{B} \neq \emptyset$.

PROOF. Let $R' = R_l \cup R_r$. Suppose by way of contradiction that $\mathcal{B} = \emptyset$. Since $S_{uniq} \neq \emptyset$, the emptiness of $\mathcal{B}$ implies that for every $X \in S_{uniq}$ there are $Y, Z \in S_{uniq}$ s.t. $(X, Y, Z) \in R'$. Thus, we may construct an infinite sequence of objects from S_{uniq}: $X_1, X_2, X_3, \ldots$, where for every $i \geq 1$ there is $Y_i \in S_{uniq}$ s.t. $(X_i, X_{i+1}, Y_i) \in R'$. Note that for every $1 \leq i < j$, $([X_i], [X_j]) \in D_M$. Since $\mathcal{O}$ is finite, then there are two equivalence classes $[X_i], [X_j]$,$1 \leq i < j$ s.t. $[X_i] = [X_j]$, implying that $([X_i], [X_i]) \in D_M$. On the other hand, constraint 5 of M implies that $D_M \cap I_\Pi = \emptyset$, which is a contradiction. ∎

Correctness of steps 4 and 5. In the following propositions we denote by $\boldsymbol{T^0}$ the initial mapping to the basic categories, and by $\boldsymbol{T^i}$ the mapping T after the i-th iteration of step 5. Note that for every $i \geq 0$, $domain(T^i) \subseteq S_{uniq}$.

PROPOSITION 4.24
For every $i \geq 0$, for every $X \in domain(T^i)$:
$T(X) \in \mathcal{B}$ iff for all $Y, Z \in S_{uniq}$ $(X, Y, Z) \notin R_l \cup R_r$;
for every $Y, Z \in S_{uniq}$:
$T^i(X) = (T^i(Y) \rightarrow T^i(Z))$ iff $(X, Y, Z) \in R_l$, and
$T^i(X) = (T^i(Z) \leftarrow T^i(Y))$ iff $(X, Y, Z) \in R_r$.

PROOF. By induction on i.

- $i = 0$:
  By step 4, $range(T^0) = \mathcal{B}$; furthermore, by step 3, for every $T(X) = \tau_X \in \mathcal{B}$, for all $Y, Z \in S_{uniq}$, $(X, Y, Z) \notin R_l \cup R_r$.
- Suppose T^i satisfies the conditions of the proposition.
  Let $X \in domain(T^{i+1} \backslash domain(T^0))$ and let $Y, Z \in S_{uniq}$. Then:
  $\Rightarrow$: if $T^{i+1}(X) = (T^{i+1}(Y) \rightarrow T^{i+1}(Z))$ then by the Category-Resolution Algorithm, in iteration $j \leq i+1$ of step 5 there was found a triple $(X, Y', Z') \in R_l$ satisfying the conditions, and $T^j(X) = (T^j(Y') \rightarrow T^j(Z'))$, where $T^i(Y') = T^i(Y)$ and $T^i(Z') = T^i(Z)$. We show that $Y = Y'$ and $Z = Z'$.
  First, note that $Y, Y', Z, Z' \in S_{uniq}$. If $T^i(Y) = T^i(Y') \in \mathcal{B}$, then $T^i(Y) = T^i(Y') = \tau_Y = \tau_{Y'}$, which implies that $Y = Y'$. Let $T^i(Y) = T^i(Y') = (T^i(W) \leftarrow T^i(U))$ (if $T^i(Y) = T^i(Y') = (T^i(U) \rightarrow T^i(W))$, the proof if similar). Then, by the induction hypothesis, $(Y, U, W) \in R_r$ and $(Y', U, W) \in R_r$. By the definition of $R_\approx$, $[Y] = [Y']$. $Y, Y' \in S_{uniq}$ implies (by proposition 4.22) that $Y = Y'$. Similarly, $Z = Z'$. Thus, $(X, Y, Z) \in R_l$.
  If $T^{i+1}(X) = (T^{i+1}(Y) \leftarrow T^{i+1}(Z))$, the proof is similar.
  $\Leftarrow$: suppose $(X, Y, Z) \in R_l$ and suppose that in iteration $j \leq i+1$ of step 5 there was found a triple $(X, Y', Z') \in R_l \cup R_r$. Constraint 2 of M implies that $(X, Y', Z') \in R_l$, and thus $T^{i+1}(X) = (T^{i+1}(Y') \rightarrow T^{i+1}(Z'))$. By the definition of $R_\approx$, $(X, Y, Z), (X, Y', Z') \in R_l$ imply that $[Y] = [Y']$ and $[Z] = [Z']$. Since $Y, Y', Z, Z' \in S_{uniq}$, proposition 4.22 implies that $Y = Y', Z = Z'$, and thus $(T^{i+1}(X) = (T^{i+1}(Y) \rightarrow T^{i+1}(Z))$.
  If $(X, Y, Z) \in R_r$, the proof is symmetric.

∎

PROPOSITION 4.25
The iteration in step 5 of the Category-Resolution Algorithm terminates.

PROOF. The iteration goes on as long as there is an $X \in (S_{uniq} \backslash domain(T))$ (satisfying the conditions specified in the algorithm). In each round one object $X \in S_{uniq}$ is given a category $T(X)$. Since S_{uniq} is finite, the iteration stops after no more than $|S_{uniq}|$ rounds. ∎

PROPOSITION 4.26
After step 5 for every $X \in S_{uniq}$:
$T(X) \in \mathcal{B}$ iff for all $Y, Z \in S_{uniq}$ $(X, Y, Z) \notin R_l \cup R_r$;
for every $Y, Z \in S_{uniq}$:
$T(X) = (T(Y) \rightarrow T(Z))$ iff $(X, Y, Z) \in R_l$, and
$T(X) = (T(Z) \leftarrow T(Y))$ iff $(X, Y, Z) \in R_r$;
for every $Y \in S_{uniq}$: $T(X) \neq T(Y)$.

PROOF. Let $R' = R_l \cup R_r$ and let i be the number of rounds of step 5. Thus, there is no triple $(X, Y, Z) \in R'$, such that $X \in (S_{uniq} \backslash domain(T^i))$ whereas $Y, Z \in domain(T^i)$. We show that $domain(T^i) = S_{uniq}$.

First, note that $domain(T^i) \subseteq S_{uniq}$. Suppose by way of contradiction that $domain(T^i) \neq S_{uniq}$, i.e., there is $X \in (S_{uniq} \backslash domain(T^i))$.

As $X \notin domain(T^i)$, then particularly $X \notin domain(T^0)$, i.e., there are $Y, Z \in S_{uniq}$ such that $(X, Y, Z) \in R'$. By the assumption, either $Y \notin domain(T^i)$ or $Z \notin domain(T^i)$. Without loss of generality, suppose $Y \notin domain(T^i)$. Note, that $([X], [Y]) \in D_M$.

Since $Y \notin domain(T^i)$, we may apply to Y the same considerations as to X above: there are $U, W \in S_{uniq}$ such that $(Y, U, W) \in R'$, and either $U \notin domain(T^i)$ or $W \notin domain(T^i)$. Without loss of generality, we suppose that $U \notin domain(T^i)$ and choose U be the next considered object, while noticing that $([Y], [U]) \in D_M$.

We may apply the above scheme infinitely many times, obtaining the infinite sequence $X_1, X_2, X_3, \ldots$ over S_{uniq}, where every pair of equivalence classes of successive objects is in the relation D_M. Since S_{uniq} is finite, then there is a pair of indices $j, k \geq 1$ $(j < k)$ such that $X_j = X_k$.

Since $([X_j], [X_{j+1}]) \in D_M, ([X_{j+1}], [X_{j+2}]) \in D_M, \ldots, ([X_{k-1}], [X_k]) \in D_M$, then $([X_j], [X_k]) \in D_M$. On the other hand, constraint 5 of M implies that $D_M \cap I_\Pi = \emptyset$, which is a contradiction.

To sum up, we obtained that $domain(T^i) = S_{uniq}$. The first two conditions of the proposition are obtained by applying proposition 4.24 to T^i, while replacing each appearance of $domain(T^i)$ with S_{uniq}.

Finally, we show that for every $X, Y \in S_{uniq}$ s.t.$X \neq Y$: $T(X) \neq T(Y)$.
Let $X, Y \in S_{uniq}$, $X \neq Y$, and suppose by way of contradiction that $T(X) = T(Y)$. There are two possibilities:

1. if $T(X) = T(Y) \in \mathcal{B}$ then, by steps 3 and 4, $T(X) = \tau_X$ and $T(Y) = \tau_Y$, thus implying that $X = Y$, which contradicts our assumption.
2. if $T(X) = T(Y) \notin \mathcal{B}$, then there are $U, W \in S_{uniq}$ such that either $T(X) = T(Y) = (T(U) \to T(W))$ or $T(X) = T(Y) = (T(W) \leftarrow T(U))$. By the first part of the above proposition, either $(X, U, W), (Y, U, W) \in R_l$ or $(X, U, W), (Y, U, W) \in R_r$. Therefore, the definition of $R_\approx$ implies that $[X] = [Y]$. Since $X, Y \in S_{uniq}$, then proposition 4.22 implies that $X = Y$, which, again, contradicts our assumption.

∎

Correctness of step 6. In the following propositions we denote by $\underline{\boldsymbol{T}}^{\mathbf{0}}$ the mapping T from S_{uniq} at the beginning of step 6, and by $\underline{\boldsymbol{T}}^{\boldsymbol{i}}$ the mapping T after the i-th iteration of step 6.

PROPOSITION 4.27
For every $i \geq 0$, for every $X \in domain(\underline{T}^i)$:

(i) $\underline{T}^i(X) \in \mathcal{B}$ iff for all $Y, Z \in \mathcal{O}$ $(X, Y, Z) \notin R_l \cup R_r$;
(ii) for all $Y, Z \in domain(\underline{T}^i)$:
$\underline{T}^i(X) = (\underline{T}^i(Y) \to \underline{T}^i(Z))$ iff $(X, Y, Z) \in R_l$, and
$\underline{T}^i(X) = (\underline{T}^i(Z) \leftarrow \underline{T}^i(Y))$ iff $(X, Y, Z) \in R_r$;
(iii) for every $Y \in domain(\underline{T}^i)$: $[X] = [Y]$ iff $\underline{T}^i(X) = \underline{T}^i(Y)$.

PROOF. By induction on i.

- $i = 0$:
Condition (i): we show that for every $X \in domain(\underline{T}^0)$: $\underline{T}^0(X) \in \mathcal{B}$ iff for all $Y, Z \in \mathcal{O}$ $(X, Y, Z) \notin R_l \cup R_r$.

$\Rightarrow$: if $T(X) \in \mathcal{B}$ then, by proposition 4.26, for all $Y, Z \in S_{uniq}$, $(X, Y, Z) \notin R_l \cup R_r$. Suppose by way of contradiction that there are $Y', Z' \in \mathcal{O}$ such that $(X, Y', Z') \in R_l \cup R_r$, where at least one of Y', Z' not in S_{uniq}. If $Y' \notin S_{uniq}$, then by proposition 4.22 there is $U \in S_{uniq}$ such that $[U] = [Y']$; if $Y' \in S_{uniq}$, then let $U = Y'$. Similarly, if $Z' \notin S_{uniq}$ then there is $W \in S_{uniq}$ such that $[W] = [Z']$; if $Z' \in S_{uniq}$, then let $W = Z'$. Since $(X, Y', Z') \in R_l \cup R_r$, then constraint 3 of M implies that $(X, U, W) \in R_l \cup R_r$, thus contradicting the fact that for all $Y, Z \in S_{uniq}$, $(X, Y, Z) \notin R_l \cup R_r$.

$\Leftarrow$: if for all $Y, Z \in \mathcal{O}$ $(X, Y, Z) \notin R_l \cup R_r$ then, in particular, for all $Y, Z \in S_{uniq}$ $(X, Y, Z) \notin R_l \cup R_r$ (since $S_{uniq} \subseteq \mathcal{O}$); thus,by proposition 4.26, $T(X) \in \mathcal{B}$.

The rest of the conditions for $\underline{T}^0$ follow directly from proposition 4.26, since $domain(\underline{T}^0) = S_{uniq}$.

- Suppose $\underline{T}^i$ satisfies the conditions of the proposition.
  - Condition (i): if $X \in domain(\underline{T}^i)$, then the condition holds by the induction hypothesis. Otherwise, there is $X' \in \mathcal{O}$ such that $X' \in domain(\underline{T}^i)$ and $\underline{T}^{i+1}(X) = \underline{T}^{i+1}(X')$, which proves the condition.
  - Condition (ii): let $X \in domain(\underline{T}^{i+1} \backslash domain(\underline{T}^0))$ and let $Y, Z \in domain(\underline{T}^i)$. If $X, Y, Z \in S_{uniq}$, then the condition holds by proposition 4.26. Otherwise, if $X, Y, Z \in domain(\underline{T}^i)$, then the condition holds by the induction hypothesis. Otherwise, suppose without loss of generality that $Y \notin domain(\underline{T}^i)$ (whereas $X, Z \in domain(\underline{T}^i)$). Then, by step 6, there is $Y' \in domain(\underline{T}^i)$ such that $(Y, Y') \in R_{\approx}$, and $\underline{T}^{i+1}(Y) = \underline{T}^i(Y')$. By the induction hypothesis:
    $\underline{T}^i(X) = (\underline{T}^i(Y') \rightarrow \underline{T}^i(Z))$ iff $(X, Y', Z) \in R_l$, and
    $\underline{T}^i(X) = (\underline{T}^i(Z) \leftarrow \underline{T}^i(Y'))$ iff $(X, Y', Z) \in R_r$.
    Furthermore, by constraint 3 of M, $(X, Y', Z) \in R_l$ iff $(X, Y, Z) \in R_l$ and, similarly, $(X, Y', Z) \in R_r$ iff $(X, Y, Z) \in R_r$. All together:
    $\underline{T}^{i+1}(X) = (\underline{T}^{i+1}(Y) \rightarrow \underline{T}^{i+1}(Z))$ iff $(X, Y, Z) \in R_l$, and
    $\underline{T}^{i+1}(X) = (\underline{T}^{i+1}(Z) \leftarrow \underline{T}^{i+1}(Y))$ iff $(X, Y, Z) \in R_r$.
  - Condition (iii): let $X, Y \in domain(\underline{T}^{i+1})$. There are three possibilities.
    - $X, Y \in domain(\underline{T}^i)$: the condition holds by induction hypothesis.
    - $X \notin domain(\underline{T}^i)$, $Y \in domain(\underline{T}^i)$: in the iteration $i+1$ there was found a pair $(X, Z) \in R_{eq}$ (and thus $[X] = [Z]$) such that $Z \in domain(\underline{T}^i)$, and T was extended to X : $\underline{T}^{i+1}(X) = \underline{T}^i(Z)$. Since $Z, Y \in domain(\underline{T}^i)$, then by the induction hypothesis $[Z] = [Y]$ iff $\underline{T}^i(Z) = \underline{T}^i(Y)$, and thus also $[X] = [Y]$ iff $\underline{T}^i(X) = \underline{T}^i(Y)$.
    - $X \in domain(\underline{T}^i)$, $Y \notin domain(\underline{T}^i)$: this case is symmetric to the previous case.

∎

PROPOSITION 4.28
The iteration in step 6 of the Category-Resolution Algorithm terminates.

PROOF. The iteration goes on as long as there is a pair $(X_1, X_2) \in R_{eq}$ (satisfying the conditions specified in the algorithm), such that $X \in (\mathcal{O} \backslash domain(T))$. In each iteration one object $X \in \mathcal{O}$ is given a category $T(X)$. Since $\mathcal{O}$ is finite, then the iteration in step 6 terminates after running no more than $|\mathcal{O}|$ rounds. ∎

Now we are ready to prove the correctness of the whole algorithm.

PROPOSITION 4.29
(Correctness of the Category-Resolution Algorithm)
The Category-Resolution Algorithm stops and returns a non-empty set of basic category $\mathcal{B}$ and a mapping $T : \mathcal{O} \longrightarrow \mathcal{C}$ such that for every $X \in \mathcal{O}$:
$T(X) \in \mathcal{B}$ iff for all $Y, Z \in \mathcal{O}$ $(X, Y, Z) \notin R_l \cup R_r$;
for every $Y, Z \in \mathcal{O}$:
$T(X) = (T(Y) \to T(Z))$ iff $(X, Y, Z) \in R_l$, and
$T(X) = (T(Z) \leftarrow T(Y))$ iff $(X, Y, Z) \in R_r$;
for every $Y \in \mathcal{O}$: $[X] = [Y]$ iff $T(X) = T(Y)$.

PROOF. Propositions 4.21, 4.25 and 4.28 imply that the Category-Resolution Algorithm terminates. Proposition 4.23 proves that $\mathcal{B} \neq \emptyset$.

Let i be the number of rounds of step 6. Thus, there is no pair $(X_1, X_2) \in R_{eq}$, such that $X_1 \notin \mathit{domain}(\underline{T}^i)$, whereas $X_2 \in \mathit{domain}(\underline{T}^i)$. We show that $\mathit{domain}(\underline{T}^i) = \mathcal{O}$.

First, note that $S_{uniq} \subseteq \mathit{domain}(\underline{T}^i) \subseteq \mathcal{O}$. Suppose by way of contradiction that $\mathit{domain}(\underline{T}^i) \neq \mathcal{O}$, i.e., there is $X \in (\mathcal{O} \setminus \mathit{domain}(\underline{T}^i))$. Since $X \notin S_{uniq}$, then by proposition 4.22 there is $Y \in S_{uniq}$ such that $[X] = [Y]$. Thus, $(X, Y) \in R_{eq}$ and $Y \in \mathit{domain}(\underline{T}^i)$, so the iteration may proceed with the pair (X, Y), which contradicts to the fact that i is the last round.

To sum up, we obtained that $\mathit{domain}(\underline{T}^i) = \mathcal{O}$. We finish the proof by applying proposition 4.27 to $\underline{T}^i$, while replacing each appearance of $\mathit{domain}(\underline{T}^i)$ with $\mathcal{O}$. ∎

4.4.2 The construction of G_M

DEFINITION 4.30
(The construction of G_M)
Let $M = (\Sigma, \mathcal{O}, X_0, \delta)$ be an **L**-automaton. Let the Category-Resolution Algorithm on M return a set of basic categories $\mathcal{B}$ and a mapping T. Construct an **L**-grammar $G_M = (\Sigma, \mathcal{B}, \tau_0, \alpha)$ as follows:

- For every $\sigma \in \Sigma$, $\alpha[\sigma] = \{T(X) \mid \text{shift}\,[\sigma, X]\}$
- $\tau_0 = T(X_0)$

The construction is well-defined by the correctness of the Category-Resolution Algorithm.

EXAMPLE 4.31
We construct the grammar G_{M_1} given the automaton M_1 from example 3.3. The output of the Category-Resolution Algorithm on M_1 is demonstrated in example 4.16. The algorithm returns a set of basic categories $\mathcal{B}$ and a mapping T. We define G_{M_1} as follows.
$G_{M_1} = (\{a, b\}, \mathcal{B}, \tau_{X_0}, \alpha)$, where:

- $\tau_{X_1} \in \alpha[a]$ because of the transition shift$[a, X_1]$;
- $((\tau_{X_1} \to \tau_{X_4}) \leftarrow \tau_{X_1}) \in \alpha[b]$ because of the transition shift$[b, X_2]$;
- $(\tau_{X_4} \to \tau_{X_0}) \in \alpha[b]$ because of the transition shift$[b, X_5]$;
- $((\tau_{X_1} \to \tau_{X_0}) \to (\tau_{X_1} \to \tau_{X_4})) \in \alpha[a]$ because of the transition shift$[a, X_7]$.

Recall the grammar G_2 from example 2.8 and notice that it is exactly our resulting grammar G_{M_1} (after renaming its basic categories: τ_{X_0} by A, τ_{X_1} by B and τ_{X_4} by C). Further we prove that every grammar G_M that is constructed from a Lambek automaton M by the above construction is tightly related to M ($M \sim G_M$); this implies that $M_1 \sim G_{M_1}$, and consequently, $M_1 \sim G_2$.

In order to prove theorem 4.15, we have to prove that $M \sim G_M$. Let $T : \mathcal{O} \longrightarrow \mathcal{C}$ be the mapping returned by the Category-Resolution Algorithm. First, we observe the following property of T.

PROPOSITION 4.32
$ST(\{\tau_0\} \cup \bigcup_{\sigma\in\Sigma} \alpha[\sigma]) \subseteq range(T)$

PROOF. In order to show this, we prove the stronger claim:

$$\text{If } \tau \in range(T), \text{ then for every } \tau' \in ST(\{\tau\}),\ \tau' \in range(T) \tag{4.1}$$

The proof is by induction on the structure of τ.

If $\tau \in \mathcal{B}$, trivial. Let $\tau \notin \mathcal{B}$. Then, either $\tau = (\tau_1 \to \tau_2)$ or $\tau = (\tau_2 \leftarrow \tau_1)$ for some τ_1, τ_2. By the Category-Resolution Algorithm, there are $X_1, X_2 \in \mathcal{O}$ such that $\tau_1 = T(X_1)$ and $\tau_2 = T(X_2)$. Let $\tau' \in ST(\{\tau\})$. Thus, $\tau' \in ST(\{\tau_1\})$ or $\tau' \in ST(\{\tau_2\})$. Therefore, $\tau' \in range(T)$ by the induction hypothesis.

Now the main claim is easily obtained.

- $(\{\tau_0\} \cup \bigcup_{\sigma\in\Sigma} \alpha[\sigma]) \subseteq range(T)$ follows directly from the definition of G_M;
- Let $\tau \in ST(\{\tau_0\} \cup \bigcup_{\sigma\in\Sigma} \alpha[\sigma])$, $\tau \notin (\{\tau_0\} \cup \bigcup_{\sigma\in\Sigma} \alpha[\sigma])$. Then $\tau \in ST(\{\tau'\})$, where $\tau' \in (\{\tau_0\} \cup \bigcup_{\sigma\in\Sigma} \alpha[\sigma])$, i.e. $\tau' \in range(T)$. $\tau \in range(T)$ follows by (4.1).

∎

The next goal is to prove that for every marked normal witness derivation Δ in G_M there is a run π of M such that $\pi \sim_T \Delta$, and in the opposite direction, for every accepting run π of M there is a marked $\mathbf{L}$-derivation Δ such that $\pi \sim_T \Delta$. We use the following propositions to prove the two directions.

PROPOSITION 4.33
Let Δ be a subtree of a marked normal witness derivation in G_M:

$$\Delta = \langle (v_1, \text{children}(v_1), \Gamma_1 \rhd \tau_1), \ldots, (v_k, \text{children}(v_k), \Gamma_k \rhd \tau_k) \rangle$$

Then, there is a run $\pi = c_{ini} \vdash^k c_k$ of M such that $\pi \sim_T \Delta$.

PROOF. First, we show that for every category τ appearing in Δ, $\tau \in range(T)$. Let Δ' be a marked normal witness derivation containing Δ, and let $\Gamma \rhd \tau_0$ be the sequent that is proved by Δ', where $\Gamma \in \alpha[w]$ for some $w \in \Sigma^+$. Then $\tau \in \Delta'$, and by the subformula property τ is a subcategory of τ_0 or of some τ_i in Γ. Since $\Gamma \in \alpha[w]$, then $\tau \in ST(\{\tau_0\} \cup \bigcup_{\sigma\in\Sigma} \alpha[\sigma])$. Proposition 4.32 implies that $\tau \in range(T)$.

We proceed by induction on k - the length of Δ.

$k = 1$: Δ is the axiom, either $\tau_1 \rhd \tau_1$ (where $\tau_1 \in \alpha[\sigma]$ for $\sigma = \text{children}(v_1)$) or $[\tau_1] \rhd \tau_1$.

- In the former case, by the definition of G_M, δ contains $shift[\sigma, X]$, where $\tau_1 = T(X)$. We choose π to be the above $shift$ transition: $\pi = \langle \sigma, \epsilon \rangle \vdash^{shift} \langle \epsilon, (X, \star) \rangle$. $\tau_1 = T(X)$ implies that $\pi \sim_T \Delta$.
- In the latter case, since $\tau_1 \in range(T)$, then there is $Y \in \mathcal{O}$ such that $\tau_1 = T(Y)$. Constraint 1 of M implies that there is $X \in [Y]$ such that $hypothesis - shift[X]$; furthermore, by the correctness of the Category-Resolution Algorithm, $T(X) = T(Y) = \tau_1$. We choose π to be the above $hypothesis - shift$ transition: $\pi = \langle \epsilon, \epsilon \rangle \vdash^{hypothesis-shift} \langle \epsilon, (X, X) \rangle$. $\tau_1 = T(X)$ implies that $\pi \sim_T \Delta$.

Suppose the proposition holds for every Δ of length $\leq k$ ($k > 1$), and let Δ be a subtree of a marked normal witness derivation tree of length $k+1$:

$$\Delta = \langle (v_1, \text{children}(v_1), \Gamma_1 \rhd \tau_1), \ldots, (v_{k+1}, \text{children}(v_{k+1}), \Gamma_{k+1} \rhd \tau_{k+1}) \rangle$$

There are three possibilities for the last inference rule applied in Δ:

- <u>arrow elimination:</u>

$$\Delta: \quad \dfrac{\Delta_1: \begin{array}{c}\vdots \\ \Gamma_i \rhd \tau_i\end{array} \quad \Delta_2: \begin{array}{c}\vdots \\ \Gamma_k \rhd \tau_k\end{array}}{\Gamma_{k+1} \rhd \tau_{k+1}}$$

  By the induction hypothesis there is a run π_1 of M such that $\pi_1 \sim_T \Delta_1$: $\pi_1 = \langle u, \epsilon \rangle \vdash^i \langle \epsilon, \mu \cdot (Y, \beta_Y) \rangle$, and there is a run π_2 of M such that $\pi_2 \sim_T \Delta_2$: $\pi_2 = \langle v, \epsilon \rangle \vdash^{k-i} \langle \epsilon, \nu \cdot (Z, \beta_Z) \rangle$, where $\tau_i = T(Y)$ and $\tau_k = T(Z)$.
  By proposition 4.3, π_1 and π_2 are balanced runs, i.e. $\mu = \nu = \epsilon$. In order to construct one run from π_1 and π_2, we 'wrap' π_1 and π_2 as follows.
  – For π_1, we add the input word v after the given input u:

$$\pi_1' = \langle uv, \epsilon \rangle \vdash^i \langle v, (Y, \beta_Y) \rangle$$

    By proposition 3.7, the resulting run π_1' has the same moves as π_1, implying that $\pi_1' \sim_T \Delta_1$.
  – For π_2, we add (Z, β_Z) at the bottom of HS before the run starts:

$$\pi_2' = \langle v, (Y, \beta_Y) \rangle \vdash^{k-i} \langle \epsilon, (Y, \beta_Y)(Z, \beta_Z) \rangle$$

    By proposition 3.7, the resulting run π_2' has the same moves as π_2, implying that $\pi_2' \sim_T \Delta_2$.
  Since the last configuration of π_1' is exactly the first configuration of π_2', π_2' may be performed after π_1'. Finally, we construct the last move for π as follows.
  Since $\tau_{k+1} \in range(T)$, then there is $X \in \mathcal{O}$ such that $\tau_{k+1} = T(X)$. Note, that either $\tau_k = (\tau_i \to \tau_{k+1})$ or $\tau_i = (\tau_{k+1} \leftarrow \tau_k)$, i.e., either $T(Z) = (T(Y) \to T(X))$ or $T(Y) = (T(X) \leftarrow T(Z)$. Thus, by the correctness of the Category-Resolution algorithm, either $left - release\,[Z, Y, X]$ or $right - release\,[Y, Z, X]$; furthermore, this implies, by constraint 4 of M, that $reduce\,[Y, Z, X]$. We construct π by first running π_1', then running π_2' and finally performing the above reduce transition:

$$\pi = \langle uv, \epsilon \rangle \vdash^i \langle v, (Y, \beta_Y) \rangle \vdash^{k-i} \langle \epsilon, (Y, \beta_Y)(Z, \beta_Z) \rangle \vdash^{reduce} \langle \epsilon, (X, \beta_X) \rangle$$

Thus, the first k steps of π correspond to the first k inference rules of Δ, as required by definition 4.1; $\tau_{k+1} = T(X)$ implies that also the last step of π corresponds to the last inference rule of Δ. Thus, $\pi \sim_T \Delta$.

- arrow introduction ($\rightarrow I$):

$$\Delta: \quad \frac{\Delta_1: \quad \dfrac{\vdots}{[\tau]\Gamma \rhd \tau_k}}{\Gamma_{k+1} \rhd \tau_{k+1}} \ (\rightarrow I) \quad (\Gamma \neq \epsilon)$$

where $\tau_{k+1} = (\tau \rightarrow \tau_k)$ and $\Gamma_{k+1} = \Gamma$. By the induction hypothesis there is a run π_1 of M such that $\pi_1 \sim_T \Delta_1$: $\pi_1 = \langle u, \epsilon\rangle \vdash^i \langle \epsilon, \mu \cdot (Y, \beta)\rangle$, where $\tau_k = T(Y)$. By proposition 4.3 π_1 is a balanced run, i.e. $\mu = \epsilon$; moreover, the same proposition implies that
– either $[\tau]\Gamma = [\tau\Gamma^l]\hat{\Gamma}'[\Gamma^r]$ $(\Gamma' \neq \epsilon)$ and $\beta = \beta^l \star \beta^r$, where $\tau\Gamma^l = T(\beta^l)$ and $\Gamma^r = T(\beta^r)$,
– or $[\tau]\Gamma = [\tau\Gamma']$ and $\tau\Gamma' = T(\beta)$.
Consequently, $\beta = Z\beta'$, where $\tau = T(Z)$. Finally, we construct the last move for π as follows. Since $\tau_{k+1} \in \mathit{range}(T)$, then there is $X \in \mathcal{O}$ such that $\tau_{k+1} = T(X)$. Note, that $\tau_{k+1} = (\tau \rightarrow \tau_k)$, i.e., $T(X) = (T(Z) \rightarrow T(Y))$. Thus, by the correctness of the Category-Resolution algorithm, $left-release\,[X, Z, Y]$. We construct π by first running π_1 and then adding the above $left-release$ transition:

$$\pi = \langle u, \epsilon\rangle \vdash^k \langle \epsilon, (Y, Z\beta')\rangle \vdash^{left-release} \langle \epsilon, (X, \beta')\rangle$$

Thus, the first k steps of π correspond to the first k inference rules of Δ, as required by definition 4.1; $\tau_{k+1} = T(X)$ implies that also the last step of π corresponds to the last inference rule of Δ. Thus, $\pi \sim_T \Delta$.

- arrow introduction ($\leftarrow I$):
Similar.

∎

PROPOSITION 4.34
Let $\pi = c_0 \vdash^k c_k$ be a subrun of an accepting run of M, such that π is balanced. Then, there is a marked **L**-derivation Δ such that $\pi \sim_T \Delta$.

PROOF. We prove the proposition by induction on k, the length of π.
$k = 1$: since π is a balanced run, it adds one element into the HS. Therefore, the single-step-balanced-run is a $shift$ or a $hypothesis - shift$ transition.

- In the former case, let the transition be $shift\,[\sigma, X]$. By the construction of G_M, $T(X) \in \alpha[\sigma]$. Let $\tau = T(X)$. We choose Δ to be the axiom $\tau \rhd \tau$ with σ at the extra level child node: $\Delta = \langle v, \sigma, \tau \rhd \tau\rangle$. Thus, $\pi \sim_T \Delta$.
- In the latter case, let the transition be $hypothesis - shift\,[X]$ and let $T(X) = \tau$. We choose Δ to be the axiom $[\tau] \rhd \tau$, implying that $\pi \sim_T \Delta$.

Suppose the proposition holds for balanced runs of length $\leq k$ $(k \geq 1)$, and let $\pi = c_0 \vdash^{k+1} c_{k+1}$ be a balanced run of length $k+1$: $\pi = \langle w, \mu\rangle \vdash^{k+1} \langle w_{k+1}, \mu \cdot (X, \beta)\rangle$. Let $\tau = T(X)$. There are five possibilities for the last move of π:

- shift: This is impossible for the balanced runs of length ≥ 2 by proposition 3.11.
- hypothesis-shift: Again, this case is impossible by proposition 3.11.
- reduce: The move $c_k \vdash c_{k+1}$ is $\langle w_k, \mu \cdot (Y, \beta_Y)(Z, \beta_Z)\rangle \vdash^{\text{reduce}} \langle w_k, \mu \cdot (X, \beta)\rangle$. Consider the prefix of π of length k: $\langle w, \mu\rangle \vdash^k \langle w_k, \mu \cdot (Y, \beta_Y)(Z, \beta_Z)\rangle$. By proposition 3.10, it may be partitioned into two balanced runs π_1 and π_2, where π_1 adds (Y, β_Y) into HS and π_2 adds (Z, β_Z) into HS:

$$\pi_1 = \langle w, \mu\rangle \vdash^i \langle w_i, \mu \cdot (Y, \beta_Y)\rangle$$
$$\pi_2 = \langle w_i, \mu \cdot (Y, \beta_Y)\rangle \vdash^{k-i} \langle w_k, \mu \cdot (Y, \beta_Y)(Z, \beta_Z)\rangle$$

By the induction hypothesis there are marked **L**-derivations Δ_1 and Δ_2, such that $\pi_1 \sim_T \Delta_1$ and $\pi_2 \sim_T \Delta_2$. Thus, $\text{root}(\Delta_1) = \Gamma_1 \rhd \tau_1$ and $\text{root}(\Delta_2) = \Gamma_2 \rhd \tau_2$, where $\tau_1 = T(Y)$, $\tau_2 = T(Z)$ and Γ_1, Γ_2 are some labeled sequences of categories. Since π is a subrun of an accepting run, then β satisfies the conditions of proposition 3.6, implying that

– $\beta = \gamma_1 \star \gamma_2$, where $\gamma_1, \gamma_2 \in \mathcal{O}^\star$

or

– $\beta \in \mathcal{O}^+$

Since also β_1, β_2 satisfy the above conditions, we have the following possibilities for the structure of β_1, β_2.

1. $\beta_1 = \beta_1^l \star \beta_1^r$ $(\beta_1^l, \beta_1^r \in \mathcal{O}^\star)$ and $\beta_2 \in \mathcal{O}^+$.
   Then, by proposition 4.3, $\Gamma_1 = [\Gamma_1^l]\hat{\Gamma}_1'[\Gamma_1^r]$ and $\Gamma_2 = [\Gamma_2']$,
   where $\Gamma_1^l = T(\beta_1^l)$, $\Gamma_1^r = T(\beta_1^r)$, and $\Gamma_2' = T(\beta_2)$. Thus, $\Gamma_1\Gamma_2 = [\Gamma_1^l]\hat{\Gamma}_1'[\Gamma_1^r\Gamma_2']$.
2. $\beta_1 = \beta_1^l \star \beta_1^r$ $(\beta_1^l, \beta_1^r \in \mathcal{O}^\star)$ and $\beta_2 \in \mathcal{O}^+$.
   Then, symmetrically, $\Gamma_1\Gamma_2 = [\Gamma_1'\Gamma_2^l]\hat{\Gamma}_2'[\Gamma_2^r]$.
3. $\beta_1 = \beta_1^l \star$ $(\beta_1^l \in \mathcal{O}^\star)$ and $\beta_2 = \star\beta_2^r$ $(\beta_2^r \in \mathcal{O}^\star)$.
   Then, by proposition 4.3, $\Gamma_1 = [\Gamma_1^l]\hat{\Gamma}_1'$ and $\Gamma_2 = \hat{\Gamma}_2'[\Gamma_2^r]$,
   where $\Gamma_1^l = T(\beta_1^l)$ and $\Gamma_2^r = T(\beta_2^r)$. Thus, $\Gamma_1\Gamma_2 = [\Gamma_1^l]\hat{\Gamma}_1'\hat{\Gamma}_2'[\Gamma_2^r]$.
4. Finally, $\beta_1, \beta_2 \in \mathcal{O}^+$. Then, by proposition 4.3, $\Gamma_1 = [\Gamma_1']$ and $\Gamma_2 = [\Gamma_2']$,
   where $\Gamma_1' = T(\beta_1)$ and $\Gamma_2' = T(\beta_2)$. Thus, $\Gamma_1\Gamma_2 = [\Gamma_1'\Gamma_2']$.

To sum up, the labeled sequence $\Gamma_1\Gamma_2$ satisfies the side condition for applying the arrow elimination rule on the reduction statements $\Gamma_1 \rhd \tau_1$ and $\Gamma_2 \rhd \tau_2$. Furthermore, since $reduce\,[Y, Z, X]$ then constraint 4 of M implies that either $left - release\,[Z, Y, X]$ or $right - release\,[Y, Z, X]$. Therefore, by the correctness of the Category-Resolution Algorithm, either $T(Z) = (T(Y) \to T(X))$ or $T(Y) = (T(X) \leftarrow T(Z))$, i.e., either $\tau_2 = (\tau_1 \to \tau)$ or $\tau_1 = (\tau \leftarrow \tau_2)$, where $\tau = T(X)$. Consequently, we may apply the arrow elimination rule on $\Gamma_1 \rhd \tau_1$ and $\Gamma_2 \rhd \tau_2$, obtaining the new marked derivation tree Δ:

$$\Delta: \quad \dfrac{\Delta_1: \begin{array}{c}\vdots\\ \Gamma_1 \rhd \tau_1\end{array} \qquad \Delta_2: \begin{array}{c}\vdots\\ \Gamma_2 \rhd \tau_2\end{array}}{\Gamma_1\Gamma_2 \rhd \tau}$$

where

– the prefix of Δ, Δ_1, satisfies $\pi_1 \sim_T \Delta_1$;

– the next part of Δ, Δ_2, satisfies $\pi_2 \sim_T \Delta_2$;

– the last inference rule of Δ is T-related to the last move of π, since $\tau = T(X)$.

Therefore, $\pi \sim_T \Delta$.

- left-release: The move $c_k \vdash c_{k+1}$ is $\langle w_k, \mu \cdot (Z, Y\beta)\rangle \vdash^{\text{left}-\text{release}} \langle w_k, \mu \cdot (X, \beta)\rangle$. Consider the prefix π_1 of π of length k: $\pi_1 = \langle w, \mu\rangle \vdash^k \langle w_k, \mu \cdot (Z, Y\beta)\rangle$. Since π is a balanced run, then $c_i = \langle w_i, \mu\mu_i\rangle$ for every $1 \leq i \leq k$, implying that π_1 is a balanced run too. By the induction hypothesis there is a marked tree Δ_1 such that $\pi_1 \sim_T \Delta_1$.
  Let $\text{root}(\Delta_1) = \Gamma_1 \rhd \tau_1$; then $\tau_1 = T(Z)$ and, by proposition 4.3,
  – either $\Gamma_1 = [\Gamma_1^l]\hat{\Gamma}_1'[\Gamma_1^r]$ $(\Gamma_1' \neq \epsilon)$ and $Y\beta = \beta_l \star \beta_r$, where $\Gamma_1^l = T(Y\beta_l)$, and $\Gamma_1^r = T(\beta_r)$
  – or $\Gamma_1 = [\Gamma_1']$ and $\Gamma_1' = T(Y\beta)$.
  Consequently, $\Gamma_1 = [\tau_2]\Gamma$ and $\tau_2 = T(Y)$. Furthermore, since $left{-}release\,[X, Y, Z]$ then, by the correctness of the Category-Resolution Algorithm, $T(X) = (T(Y) \to T(Z))$, i.e., $\tau = (\tau_2 \to \tau_1)$, where $\tau = T(X)$. Therefore, we may apply the arrow introduction rule to $[\tau_2]\Gamma' \rhd \tau_1$, thus obtaining the new marked derivation tree Δ:

$$\Delta: \quad \frac{\Delta_1: \quad \dfrac{\vdots}{[\tau_2]\Gamma \rhd \tau_1}}{\Gamma \rhd (\tau_2 \to \tau_1)}\ (\to I)$$

  where
  – the prefix of Δ, Δ_1, satisfies $\pi_1 \sim_T \Delta_1$;
  – the last inference rule of Δ is T-related to the last move of π, since $(\tau_2 \to \tau_1) = T(X)$.
  Therefore, $\pi \sim_T \Delta$.
- right-release: Similar.

∎

PROOF. **of the theorem 4.15**
We construct G_M by the construction in definition 4.30. Let $T : \mathcal{O} \longrightarrow \mathcal{C}$ be the mapping returned by the Category-Resolution algorithm during the construction of G_M. $\tau_0 = T(X_0)$ by the definition of G_M. Let Δ be a marked normal witness derivation in G_M. By proposition 4.33, there is a run $\pi = c_{ini} \vdash^k c_k$ of M such that $\pi \sim_T \Delta$. Let $\pi = c_{ini} \vdash^n c_{acc}$ be an accepting run of M. Naturally, π is a balanced run; this implies, by proposition 4.34, that there is a marked **L**-derivation Δ such that $\pi \sim_T \Delta$. ∎

Theorems 4.8 and 4.15 show that Lambek automata and **L**-grammars accept/generate the same languages, *by working in the same manner.*

5 Summary and conclusions

In this paper we have defined a computational formalism, called the 'Lambek automaton', in purpose to make it equal to **L**-grammars. Namely, **L**-automata accept the same languages as **L**-grammars do, while working in the same manner. In other words, runs of **L**-automata 'simulate' witness derivations of **L**-grammars. We have defined a relation between **L**-automata and **L**-grammars called 'tight relation', that describes

pairs automaton+grammar which assign the same structures to strings (and, consequently, whose languages coincide). Then, we have proved that every **L**-grammar is in tight relation with some **L**-automaton, and vice versa, every **L**-automaton is in tight relation with some **L**-grammar, thus concluding that an **L**-automaton is an adequate counterpart to an **L**-grammar.

A natural generalization of this work would be presentation of the 'tight relation' in a more abstract way. This can be done by defining 'tight relation' between arbitrary automata and grammar formalisms in a way independent on those formalisms. The abstract definition will relate, for example, item pushdown automata with context-free grammars (since the former 'simulate' context-free grammar derivations); similarly, it will relate **L**-automata with **L**-grammars by generalizing the relation defined in the current work. Furthermore, other combinations of grammar and automaton formalisms may be investigated in terms of satisfying 'tight relation' and new interesting formalisms may be invented while trying to associate automata with known grammar formalisms, or associate grammars with known automata families.

Another extension of this paper would be a *determinization* of an **L**-automaton. This may be done by identification of sub-families of **L**-grammars, whose corresponding **L**-automata accept their languages deterministically in a linear time by using a lookahead. This would have an important application on parsing of natural languages. This topic is currently under investigation.

References

[1] Wojciech Buszkowski. Mathematical linguistics and proof theory. In Johan van Benthem and Alice ter Muelen, editors, *Handbook of Logic and Language*, pages 683–736. North-Holland, 1997.

[2] Joachim Lambek. The mathematics of sentence structure. *Amer. Math. monthly*, 65:154–170, 1958.

[3] Michael Moortgat. Categorial type logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 93–178. North Holland, 1997.

[4] Mati Pentus. Lambek grammars are context-free. In *Proceedings of 8th IEEE symposium on Logic in Computer Science (LICS)*, pages 429 – 493, 1993.

Received October 19, 2004