Undecidability of Some Substructural Logics

Nerozhodnutelnost některých substrukturálních logik

Karel Chvalovský

Doctoral Thesis

Disertační práce

Charles University in Prague
Faculty of Arts
Department of Logic

Univerzita Karlova v Praze
Filozofická fakulta
Katedra logiky

Subject of study / Studijní obor: Logic / Logika
Supervisor / Školitelka: Marta Bílková

2015

I hereby declare that I carried out this thesis independently and only with the cited sources. This thesis is not substantially the same as any that I have submitted for a degree or diploma or other qualification at any university.

Prohlašuji, že jsem disertační práci napsal samostně s využitím pouze uvedených a řádně citovaných pramenů a literatury a že přáce nebyla využita v rámci jiného vysokoškolského studia či k získání jiného nebo stejného titulu.

In Prague / V Praze Karel Chvalovský

Abstract

This thesis deals with the algorithmic undecidability (unsolvability) of provability in some non-classical logics. In fact, there are two natural variants of this problem. Fix a logic, we can study its set of theorems or its consequence relation, which is a more general problem. It is well-known that both these problems can be undecidable already for propositional logics and we provide further examples of such logics in this thesis. In particular, we study propositional substructural logics which are obtained from the sequent calculus **LJ** for intuitionistic logic by dropping structural rules. Our main results are the following. First, (finite) consequence relations in some basic non-associative substructural logics are shown to be undecidable. Second, we prove that a basic associative substructural logic with the contraction rule, which is notorious for being hard to handle, has an undecidable set of theorems. Since the studied logics have natural algebraic semantics, we also obtain corresponding algebraic results which are interesting in their own right.

Abstrakt

Tato disertační práce se zabývá algoritmickou nerozhodnutelností (neřešitelností) dokazatelnosti v některých neklasických logikách. Ve skutečnosti existují dvě přirozené varianty toho problému. Mějme dánu logiku, pak můžeme studovat její množinu teorémů nebo její relaci důsledku, což je obecnější problém. Je známo, že oba tyto problémy mohou být nerozhodnutelné již pro výrokové logiky a tato disertační práce poskytuje další příklady takových logik. Konkrétně se věnujeme výrokovým substrukturálním logikám, které lze získat ze sekventového kalkulu **LJ** pro intuicionistickou logiku odebráním strukturálních pravidel. Naše hlavní výsledky jsou následující. Ukazujeme nerozhodnutelnost (konečné) relace důsledku pro některé základní neasociativní substrukturální logiky. Dále dokazujeme, že množina teorémů v základní substrukturální logice s pravidlem kontrakce, které obvykle způsobuje řadu komplikací, je nerozhodnutelná. Neboť studované logiky mají přirozené algebraické sémantiky, dostáváme také odpovídající algebraické výsledky, které jsou zajímavé samy o sobě.

Keywords: substructural logics, undecidability, provability, sequent calculi

Klíčová slova: substrukturální logiky, nerozhodnutelnost, dokazatelnost, sekventové kalkuly

Acknowledgments

I am deeply indebted to my colleagues whom I have been privileged to learn from. I wish some of them considered our conversations as beneficial as I did.

Let me name, in particular, Rostislav Horčík who is not only a coauthor of the second paper, but I have been very happy to discuss many topics relevant to this thesis with him.

My work has been generously supported by the Czech Science Foundation and the Grant Agency of Charles University. I was fortune to participate in projects led by Petr Cintula, Petr Hájek, and Vojtěch Kolman.

Last but not least I would like to thank my supervisor Marta Bílková for her constant support and comments on a preliminary version of the thesis.

Foreword

This thesis is comprised of an introduction and two appended papers:

1. Karel Chvalovský. 'Undecidability of Consequence Relation in Full Non-associative Lambek Calculus'. *Journal of Symbolic Logic* (to appear). (Appendix A)

2. Karel Chvalovský and Rostislav Horčík. 'Full Lambek Calculus with Contraction is Undecidable'. *Journal of Symbolic Logic* (to appear). (Appendix B)

The appendices, which are self-contained, form the core of the thesis. However, both of them are rather technical and assume some familiarity with their topics. Therefore the introduction is meant as a brief exposition for a general logician. We assume, however, that the reader has some elementary prior knowledge of sequent calculi, algorithmic undecidability, and universal algebra to mention just few topics.

Note that the papers are copyrighted by the Association for Symbolic Logic and both authors of the second paper contributed equally.

Contents

Chapter 1

Introduction

The two appendices (papers) comprising this thesis show that some natural problems in particular propositional substructural logics are algorithmically undecidable (unsolvable). Although substructural logics are a rapidly emerging field, we can hardly assume that a logician, whose interests lie outside this field, is familiar with them.

Substructural logics have various motivations that come from linguistics, computer science, philosophy etc. We are not going to discuss these motivations here, the interested reader can check many sources e.g. [Res00; Pao02; Gal+07; Bus10]. Anyway, their common feature is that we abandon classical Boolean logic for some reasons.

Say we would like to reason about words in a formal language. Moreover, for our purposes the expressive power of propositional logic seems sufficient given that we have a reasonable encoding of words, our basic data structure. One way to deal with this is to interpret letters as atoms and define the concatenation operation as a logical connective. It is quite clear that standard Boolean connectives are not suitable for that purpose.

However, we can define a new logic with connectives that have the desired properties. This does not necessarily mean that the problem is not expressible in the classical Boolean setting, but rather that there is a more convenient way how to represent it. Usually we try to avoid a complicated encoding or unnecessarily expressive formal systems, e.g. first-order logic. Therefore we study only propositional logics in this thesis. Another way how to understand this is that we have a problem expressible in a "fragment" of a classical general purpose formal system, but for some reasons we prefer to use a more problem-specific one. These reasons can be computational, philosophical, or even mere elegance can be important for us.

Clearly, this approach has its drawbacks. One of them is that we obtain a new formal system every time we change our problem. Nevertheless, it

turns out, quite interestingly, that completely different motivations often lead to the very same logic. Moreover, there is a uniform way how to obtain many of these logics using simple modifications of Gentzen's sequent calculus for classical (intuitionistic) logic. Therefore the number of systems with "reasonable" motivations seems to be relatively small. Those reasons, we believe, justify studying substructural logics.

1.1 Substructural logics and sequent calculi

Gerhard Gentzen[1] in his seminal work [Gen35a; Gen35b], which was accepted as his doctoral thesis, proposed systems of *natural deduction* for classical (**NK**[2]) and intuitionistic (**NJ**) first-order logic. One of his motivations was to set up a formal system that would more resemble actual mathematical reasoning than Hilbert (Frege) style proof systems do. In order to prove his main theorem (*Hauptsatz*)—the elimination of the cut rule—he defined sequent calculi **LK** and **LJ** as variants of systems **NK** and **NJ**, respectively.

We briefly recall some basic notions from sequent calculi for classical and intuitionistic logic, for details see e.g. [Tak87; TS00]. However, we assume that the reader is quite familiar with such calculi, because we believe it is safe to consider them folklore. Let φ, ψ, and χ be propositional formulae in a language containing conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$), and negation ($\neg$) over a set of atoms. Let Γ, Δ, Θ, and Λ be (possibly empty) finite *sequences of formulae* separated by commas. For our purposes here, it is crucial that we deal with sequences, unlike sets as usual. This is, however, the only modification we need here. The reason for that change will be clear in a moment. A *sequent* is a relation between two arbitrary finite sequences of formulae Γ and Δ, we denote it by $\Gamma \Rightarrow \Delta$ and call Γ there its *antecedent* and Δ its *succedent*. Recall that the meaning of such a sequent is that the conjunction of formulae in the antecedent implies the disjunction of formulae in the succedent ($\bigwedge \Gamma \rightarrow \bigvee \Delta$), i.e. φ is a theorem iff the sequent $\Rightarrow \varphi$ is provable.

A *proof* (or derivation) in such a calculus from a set of sequents S is a finite rooted tree with nodes labeled by sequents satisfying the following properties. Only axioms (initial sequents) or members of S can occur as leaves and all the other nodes (parents) are obtained from their children by rules that describe how new valid sequents are derived.

[1] Note that he was an associate professor at *Deutsche Karls-Universität in Prag* from 1943 until his arrest in May 1945. He died in Prague due to malnutrition after three months in prison.

[2] Note that we will use similar notation for non-associative systems later on.

The only axioms of **LK** are $\varphi \Rightarrow \varphi$ for any formula φ. On the contrary, sequent calculi have many rules which can be divided into two main groups, namely structural and logical rules. Already Gentzen[3] singled structural rules[4], which describe the structures occurring in sequents, i.e. how commas in sequents behave, out for their different nature. In **LK** the structural rules are as follows[5]

- Weakening

$$(\text{wL})\ \frac{\Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta} \qquad (\text{wR})\ \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \varphi}$$

- Contraction

$$(\text{cL})\ \frac{\varphi, \varphi, \Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta} \qquad (\text{cR})\ \frac{\Gamma \Rightarrow \Delta, \varphi, \varphi}{\Gamma \Rightarrow \Delta, \varphi}$$

- Exchange

$$(\text{eL})\ \frac{\Gamma, \varphi, \psi, \Theta \Rightarrow \Delta}{\Gamma, \psi, \varphi, \Theta \Rightarrow \Delta} \qquad (\text{eR})\ \frac{\Gamma \Rightarrow \Delta, \varphi, \psi, \Theta}{\Gamma \Rightarrow \Delta, \psi, \varphi, \Theta}$$

Here and subsequently, we tacitly assume that all rules are introduced as schemata. The left rules describe the structure of antecedents and right rules of succedents. It is easy to see that all these rules together express the fact that sequences of formulae on both sides can be seen as sets of formulae, which is the case of **LK**. Nevertheless, if we want to show that the cut rule

$$(\text{Cut})\ \frac{\Gamma \Rightarrow \Delta, \varphi \qquad \varphi, \Theta \Rightarrow \Lambda}{\Gamma, \Theta \Rightarrow \Delta, \Lambda}$$

is dispensable, Gentzen's *Hauptsatz*, then it is a clever idea to deal with sequences (or multisets) as Gentzen did. All rules mentioned so far, including the cut rule, are called structural rules in Gentzen's papers. This is a natural

[3]We should mention that he was influenced by Hertz, cf. [Došen93].

[4]Note that some Polish authors, cf. [Wój88, p. 85], use this terminology in a different context. They call rules closed under substitution as structural (or logical).

[5]We use a terminology that is prevailing in substructural logics. The order of rules respects Gentzen's papers. Gentzen's *Verdünnung* is translated in Kleene's book [Kle52] as thinning. Some people also use monotonicity, but we prefer weakening, which comes from Curry [Cur50; Cur77]. Contraction seems to be universally accepted. Exchange is a variant of interchange, following Kleene, and sometimes it is called permutation, following Curry. For further details see [Došen93].

name for they express the structural properties of proofs. However, the cut rule is, obviously, entirely different from the rest of rules. For these reasons structural rules but the cut rule are considered independently and they are called weak-structural rules in Takeuti's book [Tak87]. However, as we mainly study calculi where the cut rule is admissible[6], we call weak-structural rules simply structural rules. Hence we do not count the cut rule among structural rules, the usual convention in substructural logics.

The logical rules, which describe the behaviour of logical connectives, for **LK** are defined by

$$(\wedge\mathrm{L})\ \frac{\varphi_i, \Gamma \Rightarrow \Delta}{\varphi_1 \wedge \varphi_2, \Gamma \Rightarrow \Delta}\ \text{for } i = 1, 2 \qquad (\wedge\mathrm{R})\ \frac{\Gamma \Rightarrow \Delta, \varphi \qquad \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \wedge \psi}$$

$$(\vee\mathrm{L})\ \frac{\varphi, \Gamma \Rightarrow \Delta \qquad \psi, \Gamma \Rightarrow \Delta}{\varphi \vee \psi, \Gamma \Rightarrow \Delta} \qquad (\vee\mathrm{R})\ \frac{\Gamma \Rightarrow \Delta, \varphi_i}{\Gamma \Rightarrow \Delta, \varphi_1 \vee \varphi_2}\ \text{for } i = 1, 2$$

$$(\neg\mathrm{L})\ \frac{\Gamma \Rightarrow \Delta, \varphi}{\neg\varphi, \Gamma \Rightarrow \Delta} \qquad (\neg\mathrm{R})\ \frac{\varphi, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg\varphi}$$

$$(\to\mathrm{L})\ \frac{\Gamma \Rightarrow \Delta, \varphi \qquad \psi, \Theta \Rightarrow \Lambda}{\varphi \to \psi, \Gamma, \Theta \Rightarrow \Delta, \Lambda} \qquad (\to\mathrm{R})\ \frac{\varphi, \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \to \psi}$$

Gentzen himself noticed that the intuitionistic system **LJ** can be obtained from **LK** by assuming sequences of length at most one in succedents. Hence (cR) and (eR) become void in **LJ**. In this thesis, we concentrate solely on systems derived from **LJ**. Hence only one or no formula can occur in succedents. Although this may seem a bit restrictive, it is, in fact, a reasonable assumption, because many prominent substructural logics have this property.

Now we are ready to express what are substructural logics for us—logics described by sequent calculi obtained from **LJ** by dropping some (possibly all) structural rules. We can and will study also other structural rules (especially associativity), but the basic idea should be clear—some rules describe the structure of proofs and other rules define how logical connectives behave. We would like to keep these two groups separated. However, this requires a more careful definition of logical rules, because several variants that would be equivalent in **LJ** can easily be non-equivalent in logics lacking some structural rules. This is not particularly surprising, because the structural properties of proofs, and hence structural rules, also affect how logical connectives behave.

[6] A rule is admissible in a calculus if its set of theorems remains the same regardless of that rule.

For example, we could propose the following rules

$$(\cdot\mathrm{L})\ \frac{\Gamma, \varphi, \psi, \Delta \Rightarrow \chi}{\Gamma, \varphi \cdot \psi, \Delta \Rightarrow \chi} \qquad (\cdot\mathrm{R})\ \frac{\Gamma \Rightarrow \varphi \qquad \Delta \Rightarrow \psi}{\Gamma, \Delta \Rightarrow \varphi \cdot \psi}$$

describing a new connective called product (also called fusion). It is easy to prove that given all the structural rules as in **LJ** the connective $\cdot$ is equivalent to $\wedge$ defined before. However, if we have no structural rule this is no longer the case. In fact, under such conditions we obtain a completely new connective, which has the same properties as commas in sequents and hence can serve as the connective for the concatenation operation we looked for.

1.2 Full Lambek calculus

We have provided some warm-ups for substructural logics and indicated how to obtain them from the sequent calculus **LJ** for intuitionistic logic. Indeed, the basic system we are interested in, the *Lambek calculus* (**L**) introduced in [Lam58], is obtained from **LJ** by dropping all the previously mentioned structural rules. However, in **L** only three logical connectives are allowed, namely product ($\cdot$) and left ($\backslash$) and right ($/$) implications. It has the following (schema of) axiom and inference rules

$$(\mathrm{Id})\ \frac{}{\varphi \Rightarrow \varphi} \qquad (\mathrm{Cut})\ \frac{\Gamma, \varphi, \Delta \Rightarrow \psi \qquad \Theta \Rightarrow \varphi}{\Gamma, \Theta, \Delta \Rightarrow \psi}$$

$$(\cdot\mathrm{L})\ \frac{\Gamma, \varphi, \psi, \Delta \Rightarrow \chi}{\Gamma, \varphi \cdot \psi, \Delta \Rightarrow \chi} \qquad (\cdot\mathrm{R})\ \frac{\Gamma \Rightarrow \varphi \qquad \Delta \Rightarrow \psi}{\Gamma, \Delta \Rightarrow \varphi \cdot \psi}$$

$$(\backslash\mathrm{L})\ \frac{\Gamma, \varphi, \Delta \Rightarrow \psi \qquad \Theta \Rightarrow \chi}{\Gamma, \Theta, \chi \setminus \varphi, \Delta \Rightarrow \psi} \qquad (\backslash\mathrm{R})\ \frac{\varphi, \Gamma \Rightarrow \psi}{\Gamma \Rightarrow \varphi \setminus \psi}$$

$$(/\mathrm{L})\ \frac{\Gamma, \varphi, \Delta \Rightarrow \psi \qquad \Theta \Rightarrow \chi}{\Gamma, \varphi \,/\, \chi, \Theta, \Delta \Rightarrow \psi} \qquad (/\mathrm{R})\ \frac{\Gamma, \varphi \Rightarrow \psi}{\Gamma \Rightarrow \psi \,/\, \varphi}$$

Likely the first question to ask is why we have two implications. The reason is quite simple, we do not have the rule of exchange and hence there are (at least) two natural ways how to define this connective.

Although we motivated this whole enterprise by introducing a connective that differs significantly from standard conjunction, it can be still useful to have also connectives that more resemble standard conjunction and

disjunction. In particular, it is common to assume also lattice connectives join ($\vee$) and meet ($\wedge$) which are given by

$$(\vee\mathrm{L})\ \frac{\Gamma, \varphi, \Delta \Rightarrow \psi \qquad \Gamma, \chi, \Delta \Rightarrow \psi}{\Gamma, \varphi \vee \chi, \Delta \Rightarrow \psi} \qquad (\vee\mathrm{R})\ \frac{\Gamma \Rightarrow \varphi_i}{\Gamma \Rightarrow \varphi_1 \vee \varphi_2}\ \text{for } i = 1, 2$$

$$(\wedge\mathrm{L})\ \frac{\Gamma, \varphi_i, \Delta \Rightarrow \psi}{\Gamma, \varphi_1 \wedge \varphi_2, \Delta \Rightarrow \psi}\ \text{for } i = 1, 2 \qquad (\wedge\mathrm{R})\ \frac{\Gamma \Rightarrow \varphi \qquad \Gamma \Rightarrow \psi}{\Gamma \Rightarrow \varphi \wedge \psi}$$

The Lambek calculus with such defined join and meet is called the *Full Lambek calculus* (**FL**). However, it is more common to assume that **FL** contains also a constant 1, which corresponds to the empty sequence in antecedents, defined by

$$(1\mathrm{L})\ \frac{\Gamma, \Delta \Rightarrow \psi}{\Gamma, 1, \Delta \Rightarrow \psi} \qquad (1\mathrm{R})\ \frac{}{\Rightarrow 1}$$

Moreover, even such a logic is sometimes considered to be the positive fragment of **FL** and **FL** requires also a constant 0. We are liberal in this respect. In fact, each of two appendices uses a different convention. However, their context makes it clear whether 1 is assumed or not and the constant 0 is of no significance to us.

From our point of view, clearly, an important role is played by structural rules. In particular, we deal only with those affecting antecedents, namely *exchange* (e), *contraction* (c), and *left-weakening* or *integrality* (i) that are given by

$$(\mathrm{e})\ \frac{\Gamma, \varphi, \psi, \Delta \Rightarrow \chi}{\Gamma, \psi, \varphi, \Delta \Rightarrow \chi} \qquad (\mathrm{c})\ \frac{\Gamma, \varphi, \varphi, \Delta \Rightarrow \psi}{\Gamma, \varphi, \Delta \Rightarrow \psi} \qquad (\mathrm{i})\ \frac{\Gamma, \Delta \Rightarrow \psi}{\Gamma, \varphi, \Delta \Rightarrow \psi}$$

Structural rules are usually denoted by letters and their presence in a logic is indicated by a sequence of such letters, in a subscript, appended to the name of the original logic without them, e.g. $\mathbf{FL_c}$ is **FL** with the rule of contraction and $\mathbf{FL_{ec}}$ is **FL** with the rules of exchange and contraction.

The logics obtained from **FL** by adding a subset of those structural rules[7] play a prominent role and are called basic substructural logics in [Gal+07].

[7] In fact, also the rule *right-weakening* (o) which is given by

$$(\mathrm{o})\ \frac{\Gamma \Rightarrow}{\Gamma \Rightarrow \varphi}$$

is considered there. However, this rule plays no essential role here, because all our results hold already in positive fragments and hence also with (o). Therefore we ignore this rule completely. Moreover, it requires to allow also the empty sequence in succedents, called a stoup. Note that if we have both (i) and (o) then we call them simply *weakening* (w).

Now we should define non-associative versions of previous logics. More or less it only requires assuming trees of formulae in antecedents, unlike just sequences of formulae we assumed so far, and hence using not only commas but also brackets, e.g. the following is an example of a valid proof in the non-associative version of **L**

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{q \Rightarrow q \qquad r \Rightarrow r}{(q,r) \Rightarrow (q \cdot r)}}{p \Rightarrow p \qquad (q,r) \Rightarrow (q \cdot r)}}{(p,(q,r)) \Rightarrow (p \cdot (q \cdot r))}}{(q,r) \Rightarrow p \setminus (p \cdot (q \cdot r))}}{q \Rightarrow (p \setminus (p \cdot (q \cdot r))) \,/\, r}}{\Rightarrow q \setminus ((p \setminus (p \cdot (q \cdot r))) \,/\, r)}$$

However, a complete presentation requires more subtle formal changes, because a different approach to handle contexts is needed. For a formal definition the reader can consult Definition A.2.1 in Appendix A. Nevertheless, for the purpose of this introduction, it makes little sense to discuss these details here. The non-associative versions of **L** and **FL** are called the *Non-associative Lambek calculus* (**NL**) and the *Full Non-associative Lambek calculus* (**FNL**), respectively. Other logics are obtained from them by adding structural rules as in the associative case, e.g. $\mathbf{FNL_c}$ is **FNL** with the rule of contraction. In fact, associativity is another structural rule, even though one not considered by Gentzen. Its importance became apparent when Lambek realized that non-associative structures are more suitable for linguistic applications and with such motivations in mind he introduced **NL** in [Lam61b].

1.3 Algebraic counterparts

In the previous section, we started with a calculus having an associative product and only then we showed how to obtain the non-associative version from it. This order makes no longer sense for algebraic structures, since already the non-associative case is well-known in algebra.

An (algebraic) structure is a set with at least one finitary operation defined on it. The structures with a (non-associative) binary operation are called *groupoids* (or magmas). By abuse of notation, we use the same symbols as in the previous section to emphasize that we only obtain two different approaches to the same problem. Nevertheless, we usually follow the standard terminology, i.e. connectives are operations and formulae are terms. The previously mentioned binary operation $(\cdot)$ is again called product

or fusion. Moreover, if product is associative then such structures are called *semigroups*.

We can expand our language in numerous ways. Semigroups with a unit element (1) are *monoids* whereas we call groupoids with 1 simply *groupoids with unit*. The crucial role, for our purposes, is played by lattice operations join ($\vee$) and meet ($\wedge$), especially the former one turns out to be essential for us. Let $\langle A, \vee, \wedge \rangle$ be a lattice then its reducts $\langle A, \vee \rangle$ and $\langle A, \wedge \rangle$ are a join-semilattice and meet-semilattice, respectively. A structure $\langle A, \cdot, \vee, \wedge \rangle$ is called a *lattice-ordered groupoid* if $\langle A, \cdot \rangle$ is a groupoid, $\langle A, \vee, \wedge \rangle$ is a lattice, and product distributes over join. Similarly, we obtain (semi)lattice-ordered semigroups, monoids, and groupoids with unit. Alternatively, we can define (semi)lattices in terms of partially ordered sets—$a \leq b$ iff $a \vee b = b$ iff $a = a \wedge b$—the order is induced by the (semi)lattice structure. For simplicity of notation, we use the order as an abbreviation, e.g. using join, through the text and hence $a \leq b$ is strictly speaking an equality.

The last pair of operations we are interested in are implications which are defined by residuation laws. A *lattice-ordered residuated groupoid* $\langle A, \wedge, \vee, \cdot, \backslash, / \rangle$ is a structure such that $\langle A, \wedge, \vee \rangle$ is a lattice, $\langle A, \cdot \rangle$ is a groupoid, and for all $a, b, c \in A$ hold residuation laws:

$$a \cdot b \leq c \quad \text{iff} \quad b \leq a \backslash c \quad \text{iff} \quad a \leq c \,/\, b.$$

This explains why operations $\backslash$ and $/$ are also called divisions in algebra. Similarly, we obtain other structures—lattice-ordered residuated monoids are known as *residuated lattices* ($\mathcal{RL}$).

Recall that an identity $s = t$ holds in a class of algebras $\mathcal{K}$ if for every algebra $\mathbf{A} \in \mathcal{K}$ and every homomorphism f from terms (formulae) into the carrier set of $\mathbf{A}$ we have $f(s) = f(t)$ in $\mathbf{A}$. Hence $s \leq t$ holds in $\mathcal{K}$ if $s \vee t = t$ holds in $\mathcal{K}$.

Moreover, if $\mathcal{K}$ is a variety of algebras $\mathcal{V}$, i.e. the class of all algebras satisfying a given set of identities I, then $s = t$ holds in $\mathcal{K}$ if it is possible, roughly speaking, to obtain both t from s and s from t using the transformations allowed by I.

1.3.1 Algebraic completeness

The importance of previously defined algebras stem from the fact that they play the same role for logics we study as Boolean algebras play for classical logic, for a detailed treatment of this topic see [Gal+07]. For example, let φ

and ψ be formulae then

$$\begin{aligned} \psi \Rightarrow \varphi \text{ is provable in } \mathbf{FL} \quad &\text{iff} \quad \psi \leq \varphi \text{ holds in } \mathcal{RL}, \\ \Rightarrow \varphi \text{ is provable in } \mathbf{FL} \quad &\text{iff} \quad 1 \leq \varphi \text{ holds in } \mathcal{RL}. \end{aligned}$$

That is **FL** is complete with respect to $\mathcal{RL}$. Note that we do not distinguish corresponding operations and connectives. Hence formulae and terms are mutually exchangeable and therefore a sequent $\varphi \Rightarrow \psi$ has roughly the same meaning as the identity $\varphi \leq \psi$. Moreover, the empty sequence in an antecedent can be naturally interpreted as 1. Strictly speaking, a more complex structure than a formula or the empty structure can occur in an antecedent, however, we can equivalently express it assuming commas are interpreted as products.

In logic, we study which sequents, and therefore formulae, are provable in a given logic. The direct algebraic counterpart of this problem is the *equational theory* of a corresponding class of algebras $\mathcal{K}$, the set of all identities (equations) valid in $\mathcal{K}$.

We can easily rephrase also the provability from a set of sequents S in algebraic terms and vice versa. It suffices to restate all sequents in S as identities or the other way around. The corresponding problem, whether an identity $s = t$ holds in a class of structures $\mathcal{K}$ given a finite set of identities $E = \{s_1 = t_1, \ldots, s_n = t_n\}$, is called the *quasi-equational theory* of $\mathcal{K}$. In other words, we ask whether the quasi-identity

$$(s_1 = t_1 \text{ and } \ldots \text{ and } s_n = t_n) \text{ implies } s = t \tag{1.1}$$

holds in $\mathcal{K}$. Note that, more precisely, we have the universal closure of (1.1). In logical terms, this is equivalent to the provability of both $s \Rightarrow t$ and $t \Rightarrow s$ from $\{s_1 \Rightarrow t_1, t_1 \Rightarrow s_1, \ldots, s_n \Rightarrow t_n, t_n \Rightarrow s_n\}$. A more general problem is called the *universal theory* of $\mathcal{K}$, where any Boolean form is allowed and not only those satisfying (1.1), i.e. the set of first-order universal sentences valid in $\mathcal{K}$.

On the contrary, a more specific problem is the word problem. The *word problem* for a variety of algebras $\mathcal{V}$, given by a set of identities I, is the question whether (1.1) holds in $\mathcal{V}$ given a fixed finite set E and hence only s and t remain inputs now.[8] Roughly speaking, both transformations given by I and E are allowed. There is, however, a significant difference between I and E. Unlike all instance of transformations from I are usable, only explicit members of E are usable.

[8] We assume a finite set of generators, i.e. all terms (formulae) occurring in $E \cup \{s, t\}$ contain only variables from this set.

It is hardly surprising that structural rules we have mentioned have a very clear algebraic meaning. In sequent calculi they express properties of comma in antecedents and hence correspond to properties of product here. We have already discussed associativity. Clearly, the exchange rule is equivalent to commutativity ($x \cdot y = y \cdot x$). Similarly, the contraction and left-weakening rules correspond to the properties of being *square increasing* ($x \leq x \cdot x$) and *integral* ($x \leq 1$), respectively. Note that both these properties can be equivalently expressed as $x \wedge y \leq x \cdot y$ and $x \cdot y \leq x \wedge y$, respectively. This clarifies the role played by structural rules in the relation between product and meet.

1.4 Decidability in substructural logics

The most natural decision problem we can formulate for a logic is arguably provability in it. In fact, there are two different problems, since the deduction theorem often does not hold in substructural logics. First problem is to decide whether a formula is provable (a theorem) in the logic. Second and a more general problem, which is called the *deducibility problem* for the logic (or its finite consequence relation), is to decide whether a formula is provable in the logic from a finite set of formulae.

Many of these decision problems in substructural logics are decidable, but not all of them. In this thesis we show such undecidability results. Nevertheless, let us concentrate on decidability results first.

A common way how to prove that a substructural logic has a decidable set of theorems is using the cut elimination, the property all logics[9] we are interested in have. Assume we have a logic without the contraction rule and we want to decide whether a sequent is provable in it. This can be done easily by checking all the possible proofs of that sequent, since there are only finitely many of them.

The argument is rather straightforward. First, we can check only cut-free proofs. Second, all rules, except for the exchange rule (and associativity), have the property that sequents are always derived only from strictly "simpler" sequents. Finally, the exchange rule (and associativity) can be handled by assuming a suitable structure in antecedents, e.g. we consider multisets of formulae instead of their sequences.

[9] In logics with the contraction rule this may require a slightly modified definition of this rule; we allow whole structures to be contracted not only formulae. However, both definitions are equivalent if product is in the language. Moreover, the cut elimination is a little bit more complicated if proofs from assumptions are allowed, cf. Theorem A.2.2.

If the logic in question has the contraction rule then the situation is entirely different. The previous argument, clearly, fails, since if a sequent is not provable then there might be infinitely many conceivable proofs to check. However, it is still possible to prove the decidability of **LJ** (and **LK**) using cut-free proofs. Already Gentzen noticed that there are always only finitely many proofs which need to be checked, assuming some redundancies among them. The decidability of $\mathbf{FL_{ec}}$ is proved in [KO91] and uses the combinatorial idea developed by Kripke [Kri59] to prove that the implicational fragment of $\mathbf{FL_{ec}}$ is decidable and further extended by Meyer in his doctoral thesis [Mey66].

On the other hand, in Appendix B we show that $\mathbf{FL_c}$ is undecidable. Therefore the contraction rule can make the proof-search "essentially" infinite.

An algebraic way how to prove decidability is via the Finite Model Property (FMP). We take a class of algebras corresponding to a given logic. A class of algebras $\mathcal{K}$ has the FMP if for any identity $s = t$ that fails in $\mathcal{K}$ there exists a finite algebra $B \in \mathcal{K}$ such that $s = t$ fails in B. A finitely[10] axiomatizable class of algebras $\mathcal{K}$ with the FMP has a decidable equational theory. On the contrary, a decidable equational theory does not imply the FMP.

A stronger property than the FMP is the Finite Embeddability Property (FEP). A class of algebras $\mathcal{K}$ has the FEP if every finite subset of an $A \in \mathcal{K}$ has an isomorphic copy in a finite algebra $B \in \mathcal{K}$. Roughly speaking, the FEP is equivalent to the fact that for every quasi-identity $\bigwedge E \to s = t$, where E is a finite set of equalities, that fails in $\mathcal{K}$ there exists a finite algebra $B \in \mathcal{K}$ such that $\bigwedge E \to s = t$ fails in B. A finitely axiomatizable class of algebras $\mathcal{K}$ with the FEP has a decidable universal theory and hence also a decidable word problem. This is a standard way how to prove that the deducibility problem for a logic is decidable.

The FEP for logics with the left-weakening rule is proved in [BA05]. Note that both the associative and non-associative cases are studied there. The decidability of the deducibility problem for $\mathbf{FL_{ec}}$ follows from the fact that this logic has a form of the deduction theorem, see [Urq99]. Moreover, it is known that $\mathbf{FL_{ec}}$ has the FEP, see [Alt05], and hence it has a decidable universal theory. In the non-associative case it is known that even the deducibility problem for **NL** is decidable [Bus05] and, in fact, it is possible to prove the FEP for the fragment of **FNL** without join, see [Far08].

A brief overview of some known results, which are most relevant to our thesis, is in Table 1.1. The undecidability results mentioned there are discussed in Sections 1.6 and 1.7.

[10] Note that this property is necessary, see [Urq81].

	Theorems	**Deducibility problem**
FL	decidable	undecidable
$\mathbf{FL_e}$	decidable	undecidable
$\mathbf{FL_c}$	undecidable (see Appendix B)	undecidable
$\mathbf{FL_{ec}}$	decidable	decidable
$\mathbf{FL_i}$	decidable	decidable
FNL	decidable	undecidable (see Appendix A)
$\mathbf{FNL_e}$	decidable	undecidable (see Appendix A)
$\mathbf{FNL_c}$		undecidable (see Appendix A)
$\mathbf{FNL_{ec}}$		undecidable (see Appendix A)
$\mathbf{FNL_i}$	decidable	decidable

Table 1.1: Some substructural logics related to this thesis and their decidability.

1.5 Simple undecidable problems

Undecidability results are usually proved by encoding a problem which is already known to be undecidable into the formal system in question. There are many problems that have been shown to be undecidable since Church [Chu36b; Chu36a] and Turing [Tur37; Tur38] independently proved that there is no algorithm to solve the *Entscheidungsproblem*.[11] In this section we define some very simple formal systems which have undecidable halting problems. These systems are used in both appendices to prove our results.

We should emphasize that the systems we are about to define can compute the same problems as a universal Turing machine, but they are computationally inefficient. In fact, they were not constructed to perform real computations. Indeed, their sole purpose is to be as simple as possible but still universal computational models.

Our ultimate goal is to faithfully express such a computational system by formulae in a given logic. The systems to be defined later are particularly suitable for this purpose. However, there are further systems that has been used to prove celebrated undecidability results in logic, e.g. variants of tiling or the Post correspondence problem.

Let us emphasize that some parts of this section are more detailed and technical than necessary. Actually, even some proofs are sketched despite the fact that they are not original. Our sole motivation was to provide a deeper insight into such models of computation, because they play important roles

[11] These and other seminal papers on undecidability can be found in [Dav65].

in both appendices. However, the reader not interested in technical details is kindly asked to skip parts of this section at her or his discretion.

1.5.1 Counter machines

First, we introduce counter machines that were formalized by Minsky [Min61] and for that reason they are sometimes called after him Minsky machines. However, similar models were independently proposed by many authors. Therefore they occur in literature under different names and in many variants. In our context it is interesting to mention at least Lambek's infinite abacus [Lam61a].

Roughly speaking, they are Turing machines[12] with a different data structure and hence also instructions. Therefore they are deterministic and have finitely many states. As their title suggests, they have several unbounded counters, i.e. integers, instead of a tape. The available instructions are very simple—we can only *increment* and *decrement* these counters. However, the later instruction contains a test. If we attempt to decrement a counter which is equal to zero then it remains equal to zero but the computation continues in a different state. Hence this instruction has two possible outcomes.

Definition 1.5.1. An n-counter machine is a pair $\langle m, \tau \rangle$, where the non-zero $n \in \mathbb{N}$ is the number of its counters, the non-zero $m \in \mathbb{N}$ is the number of its states, and

$$\begin{aligned}\tau\colon [1,m] \to &\{\mathsf{INC}\} \times [1 \dots n] \times [0 \dots m] \cup \\ &\{\mathsf{DEC}\} \times [1 \dots n] \times [0 \dots m] \times [0 \dots m]\end{aligned}$$

is the program performed by the machine.

The meaning of the previous definition becomes clear from how an n-counter machine $\langle m, \tau \rangle$ computes over counters $c_1, \dots, c_n$. The computation begins in the initial state 1 by performing $\tau(1)$. The meaning of τ is as follows.

If $\tau(i) = \langle \mathsf{INC}, c, j \rangle$ then in the state i we increment the counter c_c and switch over to the state j. If $\tau(i) = \langle \mathsf{DEC}, c, j, k \rangle$ then in the state i two outcomes are possible. If the counter c_c is not equal to zero then we decrement it and switch over to the state j. Otherwise, the counter c_c is equal to zero and hence we just switch over to the state k leaving counters

[12] For our purposes, we need not care much which particular definition of a Turing machine we deal with. For example, assume we have a standard definition with an unbounded tape over a binary alphabet.

untouched. Whenever a machine reaches the final state 0 its computation ends. Therefore $\tau(0)$ is undefined.

We say that an n-counter machine $\langle m, \tau \rangle$ *accepts* an input $\langle c_1, \ldots, c_n \rangle$, an initial value of counters, if the computation of this machine over the given input ends after finitely many steps in the final state 0 with all the counters $c_1, \ldots, c_n$ equal to zero. Otherwise, we say that the machine *rejects* the input.

Example 1.5.1. We show a non-trivial computation by a 2-counter machine, how it can compute the well-known function from the Collatz conjecture ($3n + 1$ conjecture).[13] The function $\mathrm{f}\colon \mathbb{N} \to \mathbb{N}$ is given by

$$\mathrm{f}(n) = \begin{cases} n/2 & \text{if } n \text{ is even,} \\ 3n+1 & \text{if } n \text{ is odd.} \end{cases}$$

The Collatz sequence for n is the sequence n, $\mathrm{f}(n)$, $\mathrm{f}(\mathrm{f}(n))$, ... and the Collatz conjecture says that 1 occurs in the Collatz sequence for any $n > 0$.

We construct a 2-counter machine that accepts the input $\langle n, 0 \rangle$ iff 1 occurs in the Collatz sequence for n. The program is in Table 1.2. First, the machine tests whether n is equal to 1. If so it terminates. Otherwise, it computes the Collatz function (or cycle if $n = 0$) of n. It decides whether the first counter is even or odd. During this process the first counter is moved to the second counter. Finally, the value of $\mathrm{f}(n)$ is computed in the first counter and the second counter is cleared. The whole process is repeated until the first counter is equal to 1.

Naturally, our interest in these systems follows from the following theorem.

Theorem 1.5.1 ([Min61])**.** *A 2-counter machine can simulate a Turing machine.*

Idea of proof. A nice proof is based on a chain of reductions, for further details see [Min67; Sch72]. Assume we have a Turing machine with a tape over a binary alphabet. First, we can easily simulate the tape by two stacks, since the position of the machine's head halves the tape into two stacks. Second, a stack can be simulated by two counters. A stack is just a binary number and we can store it, in the reverse order, in a counter. Another counter is needed to perform the necessary operations and tests. Finally, we can easily represent finitely many counters by a counter using

[13] This function has a close connection to the decidability of some simple Turing machines and other systems, see [De 08].

State	Instruction	Meaning
1	$\langle \mathsf{DEC}, 1, 2, 17\rangle$	$n > 0$?
2	$\langle \mathsf{DEC}, 1, 3, 0\rangle$	$n = 1$?
3	$\langle \mathsf{INC}, 1, 4\rangle$	
4	$\langle \mathsf{INC}, 1, 5\rangle$	
5	$\langle \mathsf{DEC}, 1, 6, 9\rangle$	n even?
6	$\langle \mathsf{INC}, 2, 7\rangle$	
7	$\langle \mathsf{DEC}, 1, 8, 12\rangle$	n odd?
8	$\langle \mathsf{INC}, 2, 5\rangle$	
9	$\langle \mathsf{DEC}, 2, 10, 1\rangle$	$n/2$ branch
10	$\langle \mathsf{DEC}, 2, 11, 17\rangle$	
11	$\langle \mathsf{INC}, 1, 9\rangle$	
12	$\langle \mathsf{DEC}, 2, 13, 16\rangle$	$3n + 1$ branch
13	$\langle \mathsf{INC}, 1, 14\rangle$	
14	$\langle \mathsf{INC}, 1, 15\rangle$	
15	$\langle \mathsf{INC}, 1, 12\rangle$	
16	$\langle \mathsf{INC}, 1, 1\rangle$	
17	$\langle \mathsf{INC}, 1, 17\rangle$	infinite loop

Table 1.2: The Collatz sequence computed by a 2-counter machine.

a Gödel numbering and another counter is again needed to perform the necessary tests and operations, e.g. multiplying and dividing. In fact, a single counter suffices if we have multiplication and division by constants as instructions. □

Corollary 1.5.2. *The halting problem for* 2*-counter machines is undecidable.*

Remark. It is worth noting that there is no 2-counter machine that can compute 2^a from the input $\langle a, 0\rangle$, see [Sch72]. The point is that the whole coding in Theorem 1.5.1 is based on the fact that both the input and output of the "real" computation are encoded as the input of the 2-counter machine.

1.5.2 Rewriting systems

A formalism closer to logic is rewriting. We can understand the provability of $\varphi \Rightarrow \psi$ in $\mathbf{L}$ as a rewriting problem—we try to rewrite φ into ψ. For these reasons it can be useful as a middle step in proving undecidability results in

logic. There are many rewriting systems. We concentrate on string rewriting systems and their more general variant, called term rewriting systems.

String rewriting systems

A formal definition of string rewriting systems complies with the intuitive meaning suggested by their name. A string rewriting system (or semi-Thue system) is a tuple $\langle \Sigma, R \rangle$, where Σ is an alphabet and $R \subseteq \Sigma^* \times \Sigma^*$ is a binary relation on strings over Σ. A member $\langle x, y \rangle$ of R is called a (rewriting) rule and we write it $x \rightsquigarrow_R y$.

Two strings u and v are in a single-step reduction relation $\rightarrow_R \in \Sigma^* \times \Sigma^*$, we write it $u \rightarrow_R v$, if $x \rightsquigarrow_R y$, $u = sxt$, and $v = syt$ for some strings x, y, s, and t. Clearly, this generalizes to a finite-step reduction relation $\rightarrow_R^*$ which is the reflexive transitive closure of $\rightarrow_R$. The problem whether $u \rightarrow_R^* v$ holds, for a fixed v, is usually called the *reachability problem*. Note that R is usually clear from the context and hence we do not write it explicitly. For further details see e.g. [Ter03; BO93].

Example 1.5.2. We again show how to compute the Collatz sequence. Assume we have the following rewriting rules:

$$\begin{array}{ccc} q11 \rightsquigarrow 11q & 11qB \rightsquigarrow e1B & q1B \rightsquigarrow o1111b \\ & 11e \rightsquigarrow e1 & 1o \rightsquigarrow o111 \\ & Ae \rightsquigarrow Aq & Ao \rightsquigarrow Aq \end{array}$$

It holds $Aq1^nB \rightarrow^* Aq1B$ iff 1 occurs in the Collatz sequence for n. The integer n is represented by the sequence of n consecutive symbols 1 (denoted by 1^n), A and B are stoppers, and q, e, and o are pointers.

Assume we have $Aq1^nB$. The rule in the first column moves the pointer q right in steps of length two. Sooner or later this rule is no longer applicable. Then the first rule in the second or third column is used and hence n was even or odd, respectively. In the former case we replace the pointer q by e and move it left halving the number of 1's. In the later case we replace the pointer q by o and also move it left while we triple the number of 1's, the plus one part is performed during the first step. Finally, in both cases, we reach the left stopper A and then change e or o back to q.

It is hardly surprising that string rewriting systems are expressive enough to interpret 2-counter machines. The following lemma, and its proof, is sort of folklore, cf. [KS95].

Lemma 1.5.3. *A string rewriting system can simulate a 2-counter machine.*

Idea of proof. A configuration of a 2-counter machine $\langle \tau, m \rangle$ is fully determined by a triplet $\langle i, c_1, c_2 \rangle$, where i is its state and c_1 and c_2 are values of the first and second counter, respectively. This triplet can be represented by the string $Aa^{c_1}q_ib^{c_2}B$. Given such a representation it is rather straightforward how to describe the instructions of the 2-counter machine by rewriting rules. INC is easy and DEC is only slightly more complicated. Say we want to simulate an instruction $\tau(i) = \langle \mathsf{DEC}, 1, j, k \rangle$. Then the following two rules

$$aq_i \rightsquigarrow q_j$$
$$Aq_i \rightsquigarrow q_k$$

are sufficient. Note that the second rule handles the case when the first counter is equal to zero. For exactly these reasons we use stoppers A and B.

Hence the 2-counter machine accepts an input $\langle d_1, d_2 \rangle$ iff $Aa^{d_1}q_1b^{d_2}B \to^* Aq_0B$ in the string rewriting system obtained. □

Corollary 1.5.4. *The reachability problem for string rewriting systems is undecidable.*

Our main motivation to show the previous results is that the undecidability of the deducibility problem for **L** follows easily, albeit string rewriting systems are interesting in their own right. An interpretation of a string rewriting system in terms of sequent calculi is straightforward—the members of Σ are atoms and hence members of Σ^* are atoms connected by products, i.e. formulae containing only product. Clearly, all rules $x \rightsquigarrow y$ can be interpreted as sequents $x \Rightarrow y$ and a reachability problem $u \to^* v$ as the problem whether or not $u \Rightarrow v$ is provable from such a set of sequents.

Corollary 1.5.5. *The deducibility problem for the Lambek calculus* **L** *is undecidable.*

In fact, both previous corollaries follow from the following celebrated result proved independently by Post and Markov. It is not only important for our purposes here, but it is worth mentioning that it is one of the first "natural" undecidability results at all.

Theorem 1.5.6 ([Pos47; Mar47a; Mar47b])**.** *The word problem for semigroups is undecidable.*

This is, clearly, a stronger result than those following from Lemma 1.5.3, but the same proof works here, albeit equalities allow rewriting in both directions the proof is robust in this respect.

We conclude this section with Tseitin's result which is a stronger variant of the previous theorem, but it has, first and foremost, a remarkably short

presentation. Moreover, it provides an example of a seemingly easy problem presentable in plain terms which is undecidable.

Theorem 1.5.7 ([Tse58])**.** *The problem whether a word is equal to the word aaa assuming*

$$\begin{array}{lllll} ac = ca & bc = cb & eca = ce & cdca = cdcae & caaa = aaa \\ ad = da & bd = db & edb = de & & daaa = aaa \end{array}$$

is undecidable.

Term rewriting systems

It is quite obvious that more general structures than strings are required for rewriting systems to be able to deal with non-associativity or formulae (terms) in general. However, we can obtain such rewriting systems, called term rewriting systems, roughly in a similar way to the way **NL** is obtained from **L**. In fact, the objects in term rewriting systems are general terms, but it is pointless to restate all definitions for string rewriting systems from scratch because they generalize straightforwardly, for details see [Ter03]. Indeed, in string rewriting systems we replace a substring of a string by another string and in term rewriting systems we replace a subterm in a term by another term.

There is, clearly, a term rewriting system, satisfying some additional properties, which has an undecidable reachability problem, since a string rewriting system is a term rewriting system with a single binary operation satisfying associativity.

1.5.3 Tag systems

The last computational models we discuss here are tag systems. They were proposed by Post [Pos43] and again process finite words. Let $\mathcal{A} = \{a_1, \ldots, a_m\}$ be a finite alphabet of letters. A set of production rules (a program) is given by a function $\pi\colon [1, m] \to \mathcal{A}^*$, where $\mathcal{A}^*$ is the set of finite words over $\mathcal{A}$.[14] The computation of the n-tag system given by $\mathcal{A}$ and π on a word $w \in \mathcal{A}^*$ is defined as follows.

If $|w| < n$ we terminate. Otherwise, we examine the first letter in w, which is some a_i. Then we delete the first n letters in w and append $\pi(i)$ to the rest of the word after the last letter and obtain a new word.

We repeat this process until it is possible, i.e. we can run forever or at some point we terminate—we obtain a word with less than n letters. We use

[14] For convenience, we use $\pi(a_i)$ instead of $\pi(i)$ later in this section.

$\downarrow$ for such a word. In the later case we say that the n-tag system terminates on w and write $w \rightsquigarrow^*_{\mathcal{A},\pi} \downarrow$.

Example 1.5.3 ([De 08])**.** We have a 2-tag system over the alphabet $\{a, b, c\}$ with the production rules

$$\begin{aligned}\pi(a) &= bc,\\ \pi(b) &= a,\\ \pi(c) &= aaa.\end{aligned}$$

A computation of the 2-tag system on a suitable input simulates the Collatz sequence. We start with a natural number k which is encoded as a^k, a unary encoding as in string rewriting systems. The computation terminates on a^k, for $n > 0$, iff 1 is in the Collatz sequence for k. For example, the computation for $k = 3$ is

$$\begin{aligned}\underline{aaa} \to abc \to cbc \to caaa \to \underline{aaaaa} \to aaabc \to abcbc \to\\ cbcbc \to cbcaaa \to caaaaaa \to \underline{aaaaaaaa} \to aaaaaabc \to\\ aaaabcbc \to aabcbcbc \to bcbcbcbc \to bcbcbca \to bcbcaa \to\\ bcaaa \to \underline{aaaa} \to aabc \to bcbc \to bca \to \underline{aa} \to bc \to \underline{a}.\end{aligned}$$

Note that during the computation sequences of a's occur, but the computation of the Collatz sequence is shortened a bit. Whenever we should obtain a^{3l+1} from a^l, for an odd $l > 1$, we skip this step and obtain $a^{(3l+1)/2}$ immediately.

Example 1.5.4. A nice example of a 3-tag system over a binary alphabet with the rules $\pi(0) = 00$ and $\pi(1) = 1101$ was created by Post [Pos43]. However, to our knowledge, the problem whether the halting problem for this particular system is decidable remains open. For nice remarks on this example see Minsky's book [Min67, pp. 267–268]. Note that it is quite possible that this problem is very closely connected with the Collatz conjecture, see [De 09].

Theorem 1.5.8 ([CM64; Wan63])**.** *A 2-tag system can simulate a 2-counter machine.*

Idea of proof. We roughly imitate constructions from [CM64] and [Wan63] here. A configuration of a 2-counter machine $\langle m, \tau\rangle$ is given by a triplet $\langle i, c_1, c_2\rangle$, where i is its state and c_1 and c_2 are values of the first and second counter, respectively. Such a triplet can be represented by

$$q_i q_i (a_i a_i)^{2^{c_1}} (b_i b_i)^{2^{c_2}}. \tag{1.2}$$

Note that all symbols are doubled, since 2-tag systems ignore every second symbol.

Assume that $\tau(i) = \langle \mathsf{INC}, 1, j\rangle$, i.e. we increment the first counter and change the machine's state to j. This can be simulated by the following[15] production rules of a 2-tag system

$$\pi(q_i) = q_j q_j, \qquad \pi(a_i) = a_j a_j a_j a_j, \qquad \pi(b_i) = b_j b_j$$

and hence we get $q_j q_j (a_j a_j)^{2^{c_1+1}} (b_j b_j)^{2^{c_2}}$ from (1.2).

Clearly, a more complicated part is decrementing, since it contains a test. Assume that $\tau(i) = \langle \mathsf{DEC}, 1, j, k\rangle$, i.e. we increment the first counter and change the machine's state to k if $c_1 = 0$ or to j if $c_1 > 0$. We add the following rules

$$\pi(q_i) = q'_{ijk} q'_{ijk}, \qquad \pi(a_i) = a'_i, \qquad \pi(b_i) = b'_{ij} b'_{ik}.$$

Hence we obtain $q'_{ijk} q'_{ijk} (a'_i)^{2^{c_1}} (b'_{ij} b'_{ik})^{2^{c_2}}$ from (1.2). Note that if $c_1 = 0$ then there is only one a'_i which means that the whole produced word has an odd length and hence, roughly speaking, different letters are processed than if $c_1 > 0$. The computation continues using

$$\pi(q'_{ijk}) = q''_{ij} q''_{ik}, \quad \pi(a'_i) = a''_j a''_k, \qquad \pi(b'_{ij}) = b''_j b''_j, \quad \pi(b'_{ik}) = b''_k b''_k.$$

Hence we get $\cancel{q''_{ij}} q''_{ik} a''_j a''_k (b''_k b''_k)^{2^{c_2}}$ if $c_1 = 0$ or $q''_{ij} q''_{ik} (a''_j a''_k)^{2^{c_1-1}} (b''_j b''_j)^{2^{c_2}}$ if $c_1 > 0$. The decrementing is completed by applying

$$\begin{aligned} &\pi(q''_{ij}) = q_j q_j, && \pi(a''_j) = a_j a_j, && \pi(b''_j) = b_j b_j,\\ &\pi(q''_{ik}) = q_k q_k q_k, && \pi(a''_k) = a_k a_k, && \pi(b''_k) = b_k b_k. \end{aligned}$$

Therefore we get $\cancel{q_k} q_k q_k (a_k a_k)^{2^0} (b_k b_k)^{2^{c_2+1}}$ if $c_1 = 0$ or $q_j q_j (a_j a_j)^{2^{c_1-1}} (b_j b_j)^{2^{c_2}}$ if $c_1 > 0$.

It remains to show how to handle the final state q_0. For these purposes we add rules

$$\begin{aligned} &\pi(q_0) = q'_0, && \pi(q'_0) = q''_0, && \pi(q''_0) = q'''_0, && \pi(q'''_0) = q'''_0 q'''_0,\\ &\pi(a_0) = a'_0, && \pi(a'_0) = a'_0 a'_0, \\ &\pi(b_0) = b'_0 b'_0, && \pi(b'_0) = b''_0, && \pi(b''_0) = b''_0 b''_0. \end{aligned}$$

which give us $q_0 q_0 a_0 a_0 b_0 b_0 \to a_0 a_0 b_0 b_0 q'_0 \to b_0 b_0 q'_0 a'_0 \to q'_0 a'_0 b'_0 b'_0 \to b'_0 b'_0 q''_0 \to q''_0 b''_0 \to q'''_0$. □

[15] It is possible to modify these rules in order to get at most three symbols on the right side. This is, obviously, the best possible such an optimization, for details see [Wan63].

Note that 1-tag systems are decidable. It is possible to obtain an explicit upper bound, given an input and a 1-tag system, on the number of steps after which we get a repetition if the computation does not halt, for details see [Wan63].

1.6 Undecidability in substructural logics

It is, as far as we know, quite rare for a "natural" substructural logic to have an undecidable set of theorems. A well-know example, and the first one discovered, is the relevance logic **R**, which is a fragment of the involutive distributive $\mathbf{FL_{ec}}$. The undecidability of its positive fragment was proved by Urquhart [Urq84] using among other things the undecidability of the word problem for modular lattices obtained independently by Hutchinson [Hut73] and Lipshitz [Lip74].

Similarly, the variety of modular residuated lattices has an undecidable equational theory, a fact observed in [JT02], since the undecidability of the equational theory for modular lattices was proved by Freese [Fre80].

There are some other examples, the linear logic being the prominent one. In this case, however, its undecidability [Lin+92] follows from the expressive power of exponentials and, in our language, it corresponds to the fact that the deducibility problem for $\mathbf{FL_e}$ is undecidable.

The undecidability of the deducibility problem is a more widespread phenomenon. We have discussed how to prove that the deducibility problem for **L**, and hence **FL**, is undecidable using 2-counter machines. We could try to modify their encoding, see the proof of Lemma 1.5.3, to show similar results for other logics.

We have mentioned that the deducibility problem for $\mathbf{FL_e}$ is undecidable. In the setting with the exchange rule we have the following problem. A standard way to test the equality of a counter to zero is by checking whether a q_i, the symbol used to represent the state i, is next to a stopper. The exchange rule, however, can arbitrarily shuffle letters and hence that approach is no longer possible.

Nevertheless, a solution exists, it is possible to check empty counters using join. Roughly speaking, we do not rewrite a word, representing a machine's state before a given step of a computation is performed, into another word, representing a machine's state after it. Instead, in this case we rewrite a word into two words connected by join. One of them has the standard meaning, but the other one certifies that the counter is really empty. The idea to use join in this context occurred in [Lin+92] and was

simplified in [Kan95]. It is worth emphasizing that we use join in a similar fashion in both appendices, however, in a more involved way.

Another reasonable candidate for the undecidability of the deducibility problem is $\mathbf{L_c}$. Unfortunately, the contraction rule, clearly, ruins the construction, since the counters are represented by a unary encoding (sequences of letters), which can be contracted. If we know that $Aa^n q_i b^m B \Rightarrow Aq_0 B$, a standard representation of the fact that a 2-counter machine accepts the input $\langle n, m\rangle$ in the state i, is provable, then $Aa^{n'} q_i b^{m'} B \Rightarrow Aq_0 B$ is also provable, by the contraction rule, for any two numbers $0 < n' \leq n$ and $m' \leq m$.

However, there is a remedy to this problem. Obviously, if only square-free words, which contain no subword uu, could occur then the contraction rule would be inapplicable. The representation of words by their square-free variants is a well-known problem investigated already by Axel Thue, see [Lot02]. In fact, it is possible to represent a unary sequence, e.g. a^n, by a square-free sequence over an alphabet containing at least three symbols. Therefore the deducibility problem for $\mathbf{L_c}$, and hence also for $\mathbf{FL_c}$, is undecidable. The whole construction, inspired by [KS95], can be found in [Hor15], however, a brief exposition is also in Section B.3.

Recall that the deducibility problem for $\mathbf{FL_{ec}}$ is decidable. It means that it is not possible to combine previous two ideas together. Indeed, it is impossible to produce a suitable encoding which would be immune both to the rules of contraction and exchange. Similarly, it is clear that the left-weakening rule also ruins presented encodings completely, a not particularly surprising fact assuming the known decidability results for logics containing this rule.

1.7 Our results

Finally, we reached the last section of this introduction, where the results from both appendices are discussed briefly. After all, these results constitute this thesis. Although both appendices are completely independent, they share some common ideas. In particular, the way join is used in Appendix A is also essential in Appendix B.

1.7.1 Consequence relation in FNL (see Appendix A)

In Appendix A we deal with non-associative logics and show that the deducibility problem (finite consequence relation) for **FNL** is undecidable. Note that all undecidability results we have discussed so far use associativity

in an essential way. This is not a mere coincidence, the deducibility problem for **NL** is decidable and hence our results cannot be obtained solely by a more involved representation of words. Therefore lattice connectives are also necessary. In particular, the essential connective, in addition to product, is join, because it is possible to show, using the FEP, that the deducibility problem for **FNL** without join remains decidable [Far08].

Anyway, we still need to express a machine's state, given by a word, in our non-associative setting. A straightforward way is to take the word, a sequence of letters, and impose a fix convention on bracketing in it. In our case, it is convenient to assume that all brackets tight to the right side, e.g. $abcd$ is represented by $(a(b(cd)))$.

The above representation, clearly, has its drawbacks. Assume we simulate a computation and we are in a state represented by a word $ab \dots cd$ with a finite but otherwise unrestricted length. In our representation, it means we have $(a(b(...(cd)\dots)))$. If the next state of the computation is obtained, for example, by rewriting ab into e then, obviously, we have a problem. Similarly, it would be unclear how to rewrite a into ef and still keep our convention on bracketing.

It is possible, however, to overcome these problems using join, but we can simplify our situation even more by using a suitable computational model to start with. It turns out that 2-tag systems are very convenient in our situation. Roughly speaking, they require three things. First, we need to delete pairs of symbols. It is fairly easy, since deleting is the most elementary operation we can imagine here. Second, we need to append symbols. It is a routine matter to perform that, since our bracketing makes it possible to manipulate the end of a word in the same way as in the associative case. Clearly, the problem is to manipulate other parts. Finally, we need to perform previous operations correctly—the symbols to be appended entirely depend on the first symbol and the deleting and appending steps, loosely speaking, need to pass the baton between them. Therefore we need some tests to handle this, which is, clearly, the crucial part of our construction.

Roughly speaking, the use of join makes it possible to constrain the use of deleting and appending steps. We refer the reader to Appendix A for details and further technical tricks. However, an example illustrating one of techniques is appropriate.

Example 1.7.1. Let $\mathbb{S}$, a tree of formulae, be a non-associative equivalent to a context Γ, a sequence of formulae, for details see Section A.2. We assume that a sequent $(e, \mathbb{S}) \Rightarrow T \vee A$ is already provable meaning $(e, \mathbb{S})$ represents a state of a computation which terminates (T) or it is an auxiliary word (A). Indeed, the former variant holds for $(e, \mathbb{S})$ here.

Assume that a state represented by $(b, \mathbb{S})$ precedes the state represented by $(e, \mathbb{S})$ in a computation if $\mathbb{S}$ satisfies some additional properties, i.e. it is possible to rewrite b into e only under such additional conditions. Therefore we introduce an auxiliary symbol b_e and make it possible to prove $(b_e, \mathbb{S}) \Rightarrow A$ only if all conditions for rewriting b into e are fulfilled. Then, using an assumption $b \Rightarrow e \vee b_e$, we can deduce

$$\dfrac{\dfrac{\dfrac{\vdots}{(e, \mathbb{S}) \Rightarrow T \vee A} \quad \dfrac{\dfrac{\vdots}{(b_e, \mathbb{S}) \Rightarrow A}}{(b_e, \mathbb{S}) \Rightarrow T \vee A}}{(e \vee b_e, \mathbb{S}) \Rightarrow T \vee A} \qquad b \Rightarrow e \vee b_e}{(b, \mathbb{S}) \Rightarrow T \vee A}$$

Hence $b \Rightarrow e \vee b_e$ makes it possible to rewrite b into e only if we have a witness, namely b_e, certifying that all other conditions are fulfilled. It is a variant of conditional rewriting, which is used explicitly in Appendix B.

The main theorem proved in Appendix A is the following.

Theorem 1.7.1. *The deducibility problem for* **FNL** *is undecidable.*

We would like to emphasize that the whole construction works due to an interplay between product and join. Namely, it is crucial that product distributes over join. Indeed, it is really sufficient to have sequents containing only product and join and therefore such a fragment is already undecidable. Moreover, it is possible to get rid of products and use implications instead of them and hence the fragment of **FNL** with only an implication and join is also undecidable. Although it is an easy consequence of the original construction, it is not mentioned in Appendix A, because it made no sense to discuss it in the paper without going into technical details. However, the only non-trivial step required to prove this, roughly, occurs in Appendix B and hence all the necessary pieces are, more or less, available now.

Theorem 1.7.2. *The deducibility problem for the fragment of* **FNL** *containing only an implication and join is undecidable.*

Idea of proof. In a nutshell, the sequents occurring in our construction are of a very limited shape, namely

$$\varphi \Rightarrow \psi \qquad \text{or} \qquad \varphi \Rightarrow \psi \vee \chi$$

where all φ, ψ, and χ are of the form $p_1 \cdot (\cdots (p_{n-1} \cdot p_n) \cdots)$, for p_i atoms and $n > 0$. Moreover, every sequent of the form $\varphi \Rightarrow \psi \vee \chi$ can be equivalently

expressed by $\varphi \Rightarrow q_1 \vee q_2$, $q_1 \Rightarrow \psi$, and $q_2 \Rightarrow \chi$, for fresh atoms q_1 and q_2. Hence we can assume every sequent contains in the succedent a join of atoms or a formula of the form $p_1 \cdot (\cdots (p_{n-1} \cdot p_n) \cdots)$.

Any sequent containing products as the only connective in the succedent can be equivalently expressed by sequents having only an atom or a join of atoms in their succedents using the same technique as in Appendix B where string rewriting systems are transformed into atomic conditional string rewriting systems, see Section B.3.2. Although some modifications are necessary, we believe that the interested reader can deal with them.

Hence we can assume that all the sequents are of the form

$$s_1 \cdot (\cdots (s_{n-1} \cdot s_n) \cdots) \Rightarrow r_1 \vee \cdots \vee r_m$$

for $n \geq 0$ and $m > 0$ where all s_i and r_j are atoms. Any such a sequent is, obviously, equivalent to

$$\Rightarrow s_n \setminus (\ldots \setminus (s_1 \setminus (r_1 \vee \cdots \vee r_m)) \ldots),$$

which completes the proof. Clearly, it is possible to adapt the whole construction to work with / instead of $\setminus$. Roughly speaking, all brackets then tight to the left side and symbols in words encoding tag systems occur in the reverse order. □

The whole original construction is quite robust in the way that almost no modifications are required to show that the problem remains undecidable if the rule of contraction or exchange, or even both, are added. In fact, it is possible to show the same for other modifications.[16] On the contrary, if the distributive laws hold for join and meet then the deducibility problem becomes decidable [BF09]. Note that these distributive laws influence not only the provability of formulae containing join and meet, but they change also the set of formulae provable in the fragment containing only product and join. Another simple modification worth mentioning is that the one-variable fragment of **FNL** is already undecidable.

It is also possible to express our results in algebraic or term rewriting terms. Therefore we prove that some word problems and reachability problems are undecidable. It turns out that these problems are undecidable even in weaker systems than those obtained as the direct counterparts of logics we study.

[16] For example, if we have the rule called mingle which is, algebraically speaking, expressed by $xx \leq x$ or $xy \leq x \vee y$.

1.7.2 Theorems in $\mathbf{FL_c}$ (see Appendix B)

Now we confine our attention to Appendix B which deals with associative structures and a different problem. The main theorem proved in Appendix B, which is a joint work with Rostislav Horčík, is the following.

Theorem 1.7.3. *The set of theorems in* $\mathbf{FL_c}$ *is undecidable.*

In fact, it is possible to show that this already holds for the fragment of $\mathbf{FL_c}$ containing only join, meet, and an implication. Therefore already the positive fragment of $\mathbf{FL_c}$ is undecidable. On the contrary, $\mathbf{L_c}$ is decidable [Bim14, Section 9.1.5].

Our proof works as follows. We start with the string rewriting system developed to prove that the deducibility problem for $\mathbf{FL_c}$ is undecidable [Hor15], see also Section 1.6. Recall that it uses only square-free words to make the contraction rule inapplicable. We show how to express the reachability problem for that string rewriting system in terms of provability in $\mathbf{FL_c}$.

The given string rewriting system can be equivalently represented as, what we call, an atomic conditional string rewriting system. Those systems permit only letters on the right side of rewriting rules, but they allow us to restrict the use of these rules by specific contexts, cf. the use of join in Appendix A.

It is possible to encode a reachability problem in such a new system directly in $\mathbf{FL_c}$. Roughly speaking, the above-mentioned conditionality can be expressed by join, the rewriting relation corresponds to an implication, and a set of rules is a meet of their representations.

Note that the proofs of correctness and completeness of our encoding are algebraic. In fact, we show that the equational theory of square-increasing residuated lattices, which are a sound and complete algebraic semantics for (the positive fragment of) $\mathbf{FL_c}$, is undecidable. It is not essential, but such an approach seems to be more convenient than a proof-theoretical one. However, all arguments would remain the same regardless of the method used.

We can formulate an interesting corollary of our main theorem. As the set of theorems in $\mathbf{FL_c}$ is recursively enumerable but not recursive, it is possible to get the following variant of the deduction theorem. The way we obtain it is purely algorithmic and hence it differs significantly from how "standard" deduction theorems look like.

Corollary 1.7.4. *Let* $\Gamma \cup \{\varphi\}$ *be a finite set of formulae. There is an explicit algorithm that produces a formula* ψ *(given an input* φ *and* Γ*) such that* ψ *is provable in* $\mathbf{FL_c}$ *iff* φ *is provable in* $\mathbf{FL_c}$ *from* Γ*.*

Moreover, the above formula ψ, clearly, needs to contain only join, meet, and an implication.

A natural question to ask is whether or not it is possible to connect the results in both appendices and prove that the non-associative variant of $\mathbf{FL_c}$ has an undecidable set of theorems as well. Unfortunately, I have no definite answer to this problem, since I have thought about it, for several reasons, only very briefly.

Bibliography

[Alt05] Clint J. van Alten. 'The Finite Model Property for Knotted Extensions of Propositional Linear Logic'. *Journal of Symbolic Logic* 70.1 (2005), pp. 84–98. URL: http://www.jstor.org/stable/27588349.

[BA05] Willem J. Blok and Clint J. van Alten. 'On the finite embeddability property for residuated ordered groupoids'. *Transactions of the AMS* 357.10 (2005), pp. 4141–4157. DOI: 10.1090/S0002-9947-04-03654-2.

[BF09] Wojciech Buszkowski and Maciej Farulewski. 'Nonassociative Lambek Calculus with Additives and Context-Free Languages'. In: *Languages: From Formal to Natural*. Ed. by Orna Grumberg, Michael Kaminski, Shmuel Katz and Shuly Wintner. Lecture Notes in Computer Science 5533. Berlin Heidelberg: Springer, 2009, pp. 45–58. DOI: 10.1007/978-3-642-01748-3_4.

[Bim14] Katalin Bimbó. *Proof Theory: Sequent Calculi and Related Formalisms*. Discrete Mathematics and Its Applications. London: CRC Press, 2014.

[BO93] Ronald V. Book and Friedrich Otto. *String-rewriting Systems*. New York: Springer, 1993.

[Bus05] Wojciech Buszkowski. 'Lambek Calculus with Nonlogical Axioms'. In: *Language and Grammar: Studies in Mathematical Linguistics and Natural Language*. Ed. by C. Casadio, P. J. Scott and R. A. G. Seely. CSLI Lecture Notes 168. Stanford: CSLI, 2005, pp. 77–93.

[Bus10] Wojciech Buszkowski. 'Lambek Calculus and Substructural Logics'. *Linguistic Analysis* 36 (2010), pp. 15–49. URL: http://www.linguisticanalysis.com/volumes/36issue1-4.

[Chu36a] Alonzo Church. 'A Note on the Entscheidungsproblem'. *Journal of Symbolic Logic* 1.1 (1936), pp. 40–41.

[Chu36b] Alonzo Church. 'An Unsolvable Problem of Elementary Number Theory'. *American Journal of Mathematics* 58.2 (1936), pp. 345–363.

[CM64] John Cocke and Marvin Lee Minsky. 'Universality of Tag Systems with P=2'. *Journal of the ACM* 11.1 (1964), pp. 15–20. DOI: 10.1145/321203.321206.

[Cur50] Haskell Brooks Curry. *A Theory of Formal Deducibility*. Notre Dame Mathematical Lectures 6. University of Notre Dame, 1950.

[Cur77] Haskell Brooks Curry. *Foundations of Mathematical Logic*. New York: Dover Publications, 1977.

[Dav65] Martin Davis, ed. *The Undecidable: Basic Papers On Undecidable Propositions, Unsolvable Problems And Computable Functions*. New York: Raven Press, 1965.

[De 08] Lisbeth De Mol. 'Tag systems and Collatz-like functions'. *Theoretical Computer Science* 390.1 (2008), pp. 92–101. DOI: 10.1016/j.tcs.2007.10.020.

[De 09] Liesbeth De Mol. 'On the boundaries of solvability and unsolvability in tag systems. Theoretical and Experimental Results.' In: *Proceedings International Workshop on The Complexity of Simple Programs*. Ed. by Turlough Neary, Damien Woods, Tony Seda and Niall Murphy. Vol. 1. 2009, pp. 56–66. DOI: 10.4204/EPTCS.1.5.

[Došen93] Kosta Došen. 'A Historical Introduction to Substructural Logics'. In: *Substructural Logics*. Ed. by Peter Schroeder-Heister and Kosta Došen. Studies in Logic and Computation 2. Oxford: Clarendon Press, 1993.

[Far08] Maciej Farulewski. 'Finite embeddability property for residuated groupoids'. *Reports on Mathematical Logic* 43 (2008), pp. 25–42. URL: http://rml.tcs.uj.edu.pl/rml-43/02-farulewski.pdf.

[Fre80] Ralph Freese. 'Free Modular Lattices'. *Transactions of the AMS* 261.1 (1980), pp. 81–91. DOI: 10.1090/S0002-9947-1980-0576864-X.

[Gal+07] Nikolaos Galatos, Peter Jipsen, Tomasz Kowalski and Hiroakira Ono. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*. Vol. 151. Studies in Logic and the Foundations of Mathematics. Amsterdam: Elsevier, Apr. 2007, p. 532.

[Gen35a] Gerhard Gentzen. 'Untersuchungen über das logische Schließen I'. *Mathematische Zeitschrift* 39.1 (1935), pp. 176–210. DOI: 10.1007/BF01201353.

[Gen35b] Gerhard Gentzen. 'Untersuchungen über das logische Schließen II'. *Mathematische Zeitschrift* 39.1 (1935), pp. 405–431. DOI: 10.1007/BF01201363.

[Hor15] Rostislav Horčík. 'Word Problem for Knotted Residuated Lattices'. *Journal of Pure and Applied Algebra* 219.5 (May 2015), pp. 1548–1563. DOI: 10.1016/j.jpaa.2014.06.015.

[Hut73] George Hutchinson. 'Recursively unsolvable word problems of modular lattices and diagram-chasing'. *Journal of Algebra* 26.3 (1973), pp. 385–399. DOI: 10.1016/0021-8693(73)90001-X.

[JT02] Peter Jipsen and Constantine Tsinakis. 'A Survey of Residuated Lattices'. In: *Ordered Algebraic Structures:* ed. by Jorge Martínez. Vol. 7. Developments in Mathematics. Dordrecht: Kluwer, 2002.

[Kan95] Max I. Kanovich. 'The Direct Simulation of Minsky Machines in Linear Logic'. In: *Advances in Linear Logic*. Ed. by Jean-Yves Girard, Yves Lafont and Laurent Regnier. London Mathematical Society Lecture Note Series 222. Cambridge University Press, 1995.

[Kle52] Stephen Cole Kleene. *Introduction to Metamathematics*. Amsterdam: North-Holland, 1952.

[KO91] Eiji Kiriyama and Hiroakira Ono. 'The contraction rule and decision problems for logics without structural rules'. *Studia Logica* 50.2 (1991), pp. 299–319. ISSN: 0039-3215. DOI: 10.1007/BF00370189.

[Kri59] Saul Aaron Kripke. 'The problem of entailment'. *Journal of Symbolic Logic* 24 (1959). abstract, p. 324. URL: http://www.jstor.org/stable/2963903.

[KS95] Olga G. Kharlampovich and Mark V. Sapir. 'Algorithmic Problems In Varieties'. *International Journal of Algebra and Computation* 5 (1995), pp. 379–602.

[Lam58] Joachim Lambek. 'The Mathematics of Sentence Structure'. *American Mathematical Monthly* 65.3 (1958), pp. 154–170. URL: http://www.jstor.org/stable/2310058.

[Lam61a] Joachim Lambek. 'How to program an infinite abacus'. *Canadian Mathematical Bulletin* 4 (1961), pp. 295–302. DOI: 10.4153/CMB-1961-032-6.

[Lam61b] Joachim Lambek. 'On the calculus of syntactic types'. In: *Structure of Language and Its Mathematical Aspects*. Ed. by Roman Jakobson. Providence, Rhode Island: American Mathematical Society, 1961, pp. 166–178. DOI: 10.1090/psapm/012.

[Lin+92] Patrick Lincoln, John Mitchell, Andre Scedrov and Natarajan Shankar. 'Decision problems for propositional linear logic'. *Annals of Pure and Applied Logic* 56.1–3 (1992), pp. 239–311. DOI: 10.1016/0168-0072(92)90075-B.

[Lip74] Leonard Lipshitz. 'The undecidability of the word problems for projective geometries and modular lattices'. *Transactions of the AMS* 193 (1974), pp. 171–180. DOI: 10.1090/S0002-9947-1974-0364040-2.

[Lot02] M. Lothaire. *Algebraic Combinatorics on Words*. Encyclopedia of Mathematics and its Applications 90. Cambridge: Cambridge University Press, Apr. 2002.

[Mar47a] Andrei Andreevich Markov. 'Impossibility of certain algorithms in the theory of associative systems (Russian)'. *Doklady Akademii Nauk SSSR* 55 (1947), pp. 587–590.

[Mar47b] Andrei Andreevich Markov. 'Impossibility of certain algorothms in the theory of associative systems II (Russian)'. *Doklady Akademii Nauk SSSR* 58 (1947), pp. 353–356.

[Mey66] Robert K. Meyer. 'Topics in modal and many-valued logic'. PhD thesis. University of Pittsburgh, 1966.

[Min61] Marvin Lee Minsky. 'Recursive Unsolvability of Post's Problem of "Tag" and other Topics in Theory of Turing Machines'. *Annals of Mathematics* 74.3 (1961), pp. 437–455. DOI: 10.2307/1970290.

[Min67] Marvin Lee Minsky. *Computation: Finite and Infinite Machines*. London: Prentice Hall, 1967.

[Pao02] Francesco Paoli. *Substructural Logics: A Primer*. Trends in Logic. Dordrecht: Springer, 2002.

[Pos43] Emil Leon Post. 'Formal Reductions of the General Combinatorial Decision Problem'. *American Journal of Mathematics* 65.2 (1943), pp. 197–215. URL: http://www.jstor.org/stable/2371809.

[Pos47] Emil Leon Post. 'Recursive unsolvability of a problem of Thue'. *Journal of Symbolic Logic* 12.1 (Mar. 1947), pp. 1–11. DOI: 10.2307/2267170.

[Res00] Greg Restall. *An Introduction to Substructural Logics*. London: Routledge, 2000.

[Sch72] Rich Schroeppel. *A two counter machine cannot calculate* 2^N. Tech. rep. A. I. Laboratory, M.I.T., 1972.

[Tak87] Gaisi Takeuti. *Proof Theory*. 2nd. Oxford: North-Holland, 1987.

[Ter03] Terese, ed. *Term Rewriting Systems*. Cambridge Tracts in Theoretical Computer Science 55. Cambridge: Cambridge University Press, 2003.

[TS00] Anne Sjerp Troelstra and Helmut Schwichtenberg. *Basic Proof Theory*. 2nd. Cambridge Tracts in Theoretical Computer Science 43. Cambridge University Press, July 2000.

[Tse58] Grigorii Samuilovich Tseitin. 'An associative calculus with an insoluble problem of equivalence'. *Trudy Matematicheskogo Instituta im. V. A. Steklova* 52 (1958), pp. 172–189. URL: http://mi.mathnet.ru/eng/tm1317.

[Tur37] Alan Mathison Turing. 'On Computable Numbers, with an Application to the Entscheidungsproblem'. *Proceedings of London Mathematical Society* s2-42.1 (1937), pp. 230–265. DOI: 10.1112/plms/s2-42.1.230.

[Tur38] Alan Mathison Turing. 'On Computable Numbers, with an Application to the Entscheidungsproblem. A Correction'. *Proceedings of London Mathematical Society* s2-43.1 (1938), pp. 544–546. DOI: 10.1112/plms/s2-43.6.544.

[Urq81] Alasdair Urquhart. 'Decidability and the Finite Model Property'. *Journal of Philosophical Logic* 10.3 (1981), pp. 367–370. URL: http://www.jstor.org/stable/30226231.

[Urq84] Alasdair Urquhart. 'The Undecidability of Entailment and Relevant Implication'. *Journal of Symbolic Logic* 49.4 (1984), pp. 1059–1073. DOI: 10.2307/2274261.

[Urq99] Alasdair Urquhart. 'The Complexity of Decision Procedures in Relevance Logic II'. *Journal of Symbolic Logic* 64.4 (1999), pp. 1774–1802. DOI: 10.2307/2586811.

[Wan63] Hao Wang. 'Tag systems and lag systems'. *Mathematische Annalen* 152.1 (1963), pp. 65–74. DOI: 10.1007/BF01343730.

[Wój88] Ryszard Wójcicki. *Theory of Logical Calculi: Basic Theory of Consequence Operation*. Vol. 199. Synthese Library. Dordrecht: Kluwer, 1988.

Appendix A

Undecidability of Consequence Relation in Full Non-associative Lambek Calculus[†]

Karel Chvalovský

Institute of Computer Science, Academy of Sciences of the Czech Republic,
Pod Vodárenskou věží 271/2, 182 07 Prague 8, Czech Republic

Abstract

We prove that the consequence relation in the Full Non-associative Lambek Calculus is undecidable. An encoding of the halting problem for 2-tag systems using finitely many sequents in the language $\{\cdot, \vee\}$ is presented. Therefore already the consequence relation in this fragment is undecidable. Moreover, the construction works even when the structural rules of exchange and contraction are added.

A.1 Introduction

There are many motivations for studying substructural logics, which can be seen as logics lacking some structural rules when presented in the form of sequent calculi. It is of no surprise that some of the motivations come from linguistics. Lambek [Lam58] introduced a calculus, which is now called after him (**L**), with exactly these motivations in mind. All the normal structural rules, i.e. exchange, contraction, and weakening, are missing in **L**. The

[†]Karel Chvalovský. 'Undecidability of Consequence Relation in Full Non-associative Lambek Calculus'. *Journal of Symbolic Logic* (to appear).

standard language of **L** contains product ($\cdot$) and two implications ($/$ and $\backslash$) as connectives. Nevertheless, it is common in substructural logics to consider also additive join ($\vee$) and meet ($\wedge$). By adding rules for all these connectives we obtain the Full Lambek Calculus (**FL**).

With linguistics in mind it is natural that Lambek [Lam61] also introduced a non-associative variant of his calculus, we call it **NL**. The main difference can be demonstrated by structures used to represent sequents. In **L** the basic structure is a sequence of formulae, but non-associativity in **NL** requires a tree with leaves that are labelled by formulae. Again by adding rules for join and meet we obtain the Full Non-associative Lambek Calculus (**FNL**), for details see e.g. [BF09; GO10].

A natural question to ask is whether or not the provability in these calculi is decidable. It is known [BF09] that provability in **FNL** is decidable in polynomial space. The problem we deal with in this paper is whether provability in **FNL** is decidable if finitely many non-logical axioms are added, i.e. we study the decidability of the finitary consequence relation in **FNL**. Although this problem was shown [Bus05] to be decidable in polynomial time for **NL**, we prove that it is undecidable for **FNL** by encoding the halting problem for 2-tag systems in the language $\{\cdot, \vee\}$, i.e. we show that the provability from finitely many non-logical axioms is undecidable in this fragment. Moreover, it is undecidable even if the structural rules of exchange and contraction are added. It is worth noting that we are in non-associative setting, because associativity usually plays an important role in similar results, cf. the undecidability of consequence relation in **L** [Bus82; Bus05]. However, we show that the distributivity of product over join is sufficient. It is worth noting that the consequence relation in the distributive **FNL**, the distributive laws hold for $\{\wedge, \vee\}$, is decidable, see [BF09; Far08; HH14].

Although this paper deals solely with logic, there are some algebraic consequences. **FNL** is complete with respect to lattice-ordered residuated groupoids. Therefore we prove that their word problem is undecidable. This solves negatively Problem 7.1 in [HH14]. Moreover, the encoding of the halting problem for 2-tag systems can be used directly to obtain a similar result for their $\{\cdot, \vee\}$-reduct—join-semilattices expanded by a groupoid operation (product) where product distributes over join. Our construction shows that the word problem for such structures is generally undecidable. This is true even if $x \cdot y = y \cdot x$ and $x \leq x \cdot x$ hold. Moreover, we do not need the idempotency and commutativity of join in full generality.

The paper is organized as follows. The next section contains some basic definitions. As for our purposes it is sufficient to deal with $\{\cdot, \vee\}$ we concentrate solely on this language. Nevertheless, we present an entire sequent calculus for **FNL** first. Then we discuss a sequent calculus for

the $\{\cdot, \vee\}$-fragment and some equivalences on formulae. In the last part of this section tag systems are introduced. In Section A.3 we describe how a tag system can be encoded in the language $\{\cdot, \vee\}$. The most important part describes non-logical axioms which express the behaviour of a given tag system. Then we prove, in Sections A.4 and A.5, the correctness and completeness of our encoding. In Section A.6 we discuss some possible modifications—addition of the structural rules and the one-variable fragment. Section A.7 contains some algebraic consequences of our construction. A formulation of our results in terms of term rewriting systems is briefly discussed in Section A.8.

A.2 Preliminaries

Formulae are formed out in the standard way from a denumerable set of propositional variables (atoms) using connectives and parentheses. All the connectives are binary: product ($\cdot$), two implications ($\backslash$ and $/$), join ($\vee$), and meet ($\wedge$). We denote formulae by small Greek letters and do not write parentheses if no confusion can arise.

As we are in non-associative setting we have to impose some other notions, following e.g. [Bus05; BF09; GO10], which simplify the formulation of rules in our sequent calculus and handling of substitutions later on. We say that all formulae are (atomic) *structures*, the empty structure ε is a structure, and if $\mathbb{S}$ and $\mathbb{T}$ are structures then $(\mathbb{S}, \mathbb{T})$ is a structure, nothing else is a structure. Moreover, $(\varepsilon, \mathbb{S}) = (\mathbb{S}, \varepsilon) = \mathbb{S}$ and we always assume that a structure does not contain unnecessary empty structures. A *context* is a structure $\mathbb{S}[\circ]$ which has a single occurrence of the special structure symbol $\circ$, a place for substitution. Then $\mathbb{S}[\mathbb{T}]$ denotes the result of substitution of $\mathbb{T}$ for $\circ$ in $\mathbb{S}[\circ]$. We use the blackboard bold font for structures and contexts are distinguished by using square brackets for substitutions.

A.2.1 Full Non-associative Lambek Calculus

We present a sequent calculus for **FNL**, cf. [Bus05; BF09; GO10]. For any structure $\mathbb{S}$ and formula φ we say that $\mathbb{S} \Rightarrow \varphi$ is a *sequent*. The definition of provability in a sequent calculus is standard—a proof is a tree labelled by sequents, where only axioms can occur as leaves and every other vertex (sequent) is obtained by an inference rule from its children.

Definition A.2.1. Let φ, ψ, χ, φ_1, and φ_2 be arbitrary formulae and $\mathbb{S}$, $\mathbb{T}$ be arbitrary structures. The sequent calculus for **FNL** has the following axioms and inference rules:

$$\text{(Id)}\ \varphi \Rightarrow \varphi \qquad \text{(Cut)}\ \frac{\mathbb{S}[\varphi] \Rightarrow \psi \qquad \mathbb{T} \Rightarrow \varphi}{\mathbb{S}[\mathbb{T}] \Rightarrow \psi}$$

$$(\cdot\text{L})\ \frac{\mathbb{S}[(\varphi, \psi)] \Rightarrow \chi}{\mathbb{S}[\varphi \cdot \psi] \Rightarrow \chi} \qquad (\cdot\text{R})\ \frac{\mathbb{S} \Rightarrow \varphi \qquad \mathbb{T} \Rightarrow \psi}{(\mathbb{S}, \mathbb{T}) \Rightarrow \varphi \cdot \psi}$$

$$(\backslash\text{L})\ \frac{\mathbb{S}[\varphi] \Rightarrow \psi \qquad \mathbb{T} \Rightarrow \chi}{\mathbb{S}[(\mathbb{T}, \chi\backslash\varphi)] \Rightarrow \psi} \qquad (\backslash\text{R})\ \frac{(\varphi, \mathbb{S}) \Rightarrow \psi}{\mathbb{S} \Rightarrow \varphi\backslash\psi}$$

$$(/\text{L})\ \frac{\mathbb{S}[\varphi] \Rightarrow \psi \qquad \mathbb{T} \Rightarrow \chi}{\mathbb{S}[(\varphi/\chi, \mathbb{T})] \Rightarrow \psi} \qquad (/\text{R})\ \frac{(\mathbb{S}, \varphi) \Rightarrow \psi}{\mathbb{S} \Rightarrow \psi/\varphi}$$

$$(\vee\text{L})\ \frac{\mathbb{S}[\varphi] \Rightarrow \chi \qquad \mathbb{S}[\psi] \Rightarrow \chi}{\mathbb{S}[\varphi \vee \psi] \Rightarrow \chi} \qquad (\vee\text{R})\ \frac{\mathbb{S} \Rightarrow \varphi_i}{\mathbb{S} \Rightarrow \varphi_1 \vee \varphi_2}\ \text{for } i = 1, 2$$

$$(\wedge\text{L})\ \frac{\mathbb{S}[\varphi_i] \Rightarrow \psi}{\mathbb{S}[\varphi_1 \wedge \varphi_2] \Rightarrow \psi}\ \text{for } i = 1, 2 \qquad (\wedge\text{R})\ \frac{\mathbb{S} \Rightarrow \varphi \qquad \mathbb{S} \Rightarrow \psi}{\mathbb{S} \Rightarrow \varphi \wedge \psi}$$

Our aim is to discuss calculi with non-logical axioms. Therefore we extend the previous definition to handle them. They are sequents $\mathbb{S} \Rightarrow \alpha$, where $\mathbb{S}$ and α are a structure and a formula, respectively. However, let us remark that non-logical axioms are not closed under substitutions and therefore propositional variables in them are treated as constants. We use Φ for a *set of non-logical axioms* and $\mathbf{FNL}(\Phi)$ for the sequent calculus containing axioms and rules for **FNL** and non-logical axioms from Φ.

As the reader probably anticipated the intended meaning of comma in structures is product. Two natural extremes arise: the case when there are as many as possible commas instead of products and vice versa. For this purposes we define translations σ and ρ.

Definition A.2.2. Let $\mathbb{S}$ be any structure. The functions σ and ρ on structures are given by

$$\sigma(\mathbb{S}) = \begin{cases} (\sigma(\mathbb{T}), \sigma(\mathbb{U})) & \text{if } \mathbb{S} = (\mathbb{T}, \mathbb{U}), \\ (\sigma(\varphi), \sigma(\psi)) & \text{if } \mathbb{S} = \varphi \cdot \psi, \\ \mathbb{S} & \text{otherwise.} \end{cases}$$

$$\rho(\mathbb{S}) = \begin{cases} \rho(\mathbb{T}) \cdot \rho(\mathbb{U}) & \text{if } \mathbb{S} = (\mathbb{T}, \mathbb{U}), \\ \mathbb{S} & \text{otherwise.} \end{cases}$$

It is clear that $\rho(\mathbb{S}) = \rho(\sigma(\mathbb{S}))$, $\sigma(\mathbb{S}) = \sigma(\rho(\mathbb{S}))$, and $\rho(\mathbb{S})$ is a formula, for non-empty $\mathbb{S}$.

Example A.2.1. It holds that $\sigma(p \cdot (q \cdot r)) = (p, (q, r))$, but $\sigma(p \vee (q \cdot r)) = p \vee (q \cdot r)$, where p, q, and r are atoms.

Lemma A.2.1. *For any set of non-logical axioms Φ, structure $\mathbb{S}$, and formula φ it holds that $\mathbb{S} \Rightarrow \varphi$ is provable in* $\mathbf{FNL}(\Phi)$ *iff $\sigma(\mathbb{S}) \Rightarrow \varphi$ is provable in* $\mathbf{FNL}(\Phi)$ *iff $\rho(\mathbb{S}) \Rightarrow \varphi$ is provable in* $\mathbf{FNL}(\Phi)$.

Proof. It is sufficient to use $(\cdot\mathrm{R})$ and (Cut), or $(\cdot\mathrm{L})$. □

As our aim is to produce an encoding of an undecidable problem, we will have to show that such an encoding is correct and complete. In order to prove completeness we will study all the possible proofs of some sequents in our calculus. For this reason we want to produce a simpler calculus which proves the very same sequents.

A natural restriction is to allow only very specific cuts. We call a sequent $\mathbb{S} \Rightarrow \varphi$ *regular* if the only formulae occurring in the structure $\mathbb{S}$ are propositional variables, i.e. it contains no connectives. A set of non-logical axioms Φ is regular if all its members are regular. Let us assume the notation in Definition A.2.1 for (Cut). We call φ there the *cut formula*. We say that a cut is *principal* if $\mathbb{T} \Rightarrow \varphi$ is a regular sequent from non-logical axioms, cf. [Tro92, p. 174]. A proof containing only principal cuts is called *standard*.

We need to define some more notions. The *grade of a cut* is the length of the cut formula φ. The *size of a cut* is the number of sequents appearing in the proof of $\mathbb{S}[\varphi] \Rightarrow \psi$ and $\mathbb{T} \Rightarrow \varphi$. However, the size of all principal cuts is defined to be 0. Every application of a rule for connectives creates a formula and we call it the *main formula*. Other formulae appearing in this sequent are called *side formulae*.

Theorem A.2.2. *For any regular set of non-logical axioms Φ, structure $\mathbb{S}$, and formula φ it holds that $\mathbb{S} \Rightarrow \varphi$ has a proof in* $\mathbf{FNL}(\Phi)$ *iff $\mathbb{S} \Rightarrow \varphi$ has a standard proof in* $\mathbf{FNL}(\Phi)$.

Proof. This is proved by standard cut-elimination techniques, cf. [Tro92; Gal+07; Došš88; Ono98]. The proof is by double induction on the grade and size of cuts. We take a top most (closest to leaves) non-principal cut and transform it into another cut and the sum of grades of all cuts in the proof either decreases, or remains equal and then the sum of their sizes decreases. As this process is well-founded it suffices to show that we are able to transform every possible non-principal cut.

Let us assume we have a top most non-principal cut. If the cut formula is a side formula in any of two input sequents then we can apply the rule for a sequent where the cut formula is a side formula after the cut. Therefore we obtain a cut with the same grade but smaller size.

If both the main formulae in the input sequents are the same as the cut formula then both were obtained by rules for the same connective. Let us assume that it was e.g. $\backslash$. Then we can transform

$$\dfrac{\dfrac{\dfrac{\vdots}{\mathbb{T}[\xi] \Rightarrow \psi} \quad \dfrac{\vdots}{\mathbb{U} \Rightarrow \chi}}{\mathbb{T}[(\mathbb{U}, \chi\backslash\xi)] \Rightarrow \psi}\,(\backslash\mathrm{L}) \quad \dfrac{\dfrac{\vdots}{(\chi, \mathbb{V}) \Rightarrow \xi}}{\mathbb{V} \Rightarrow \chi\backslash\xi}\,(\backslash\mathrm{R})}{\mathbb{T}[(\mathbb{U}, \mathbb{V})] \Rightarrow \psi}\,(\mathrm{Cut})$$

into

$$\dfrac{\dfrac{\dfrac{\vdots}{\mathbb{T}[\xi] \Rightarrow \psi} \quad \dfrac{\vdots}{(\chi, \mathbb{V}) \Rightarrow \xi}}{\mathbb{T}[(\chi, \mathbb{V})] \Rightarrow \psi}\,(\mathrm{Cut}) \quad \dfrac{\vdots}{\mathbb{U} \Rightarrow \chi}}{\mathbb{T}[(\mathbb{U}, \mathbb{V})] \Rightarrow \psi}\,(\mathrm{Cut})$$

and we obtain two cuts with the sum of grades smaller then in the original cut. Similarly for other connectives.

As we have a set Φ we must handle the cases where a cut is applied on other principal cut(s). In such a case we can always transform

$$\dfrac{\dfrac{\vdots}{\mathbb{T}[\xi] \Rightarrow \psi} \quad \dfrac{\dfrac{\dfrac{\vdots}{\mathbb{U}[\chi] \Rightarrow \xi} \quad \mathbb{V} \Rightarrow \chi}{\mathbb{U}[\mathbb{V}] \Rightarrow \xi}\,(\mathrm{Cut})}{\mathbb{T}[\mathbb{U}[\mathbb{V}]] \Rightarrow \psi}\,(\mathrm{Cut})$$

into

$$\dfrac{\dfrac{\dfrac{\vdots}{\mathbb{T}[\xi] \Rightarrow \psi} \quad \dfrac{\vdots}{\mathbb{U}[\chi] \Rightarrow \xi}}{\mathbb{T}[\mathbb{U}[\chi]] \Rightarrow \psi}\,(\mathrm{Cut}) \quad \mathbb{V} \Rightarrow \chi}{\mathbb{T}[\mathbb{U}[\mathbb{V}]] \Rightarrow \psi}\,(\mathrm{Cut})$$

where we can assume $\mathbb{V} \Rightarrow \chi$ is a member of Φ. Therefore we obtain cuts with smaller size and zero size while the sum of grades remains the same.

The final non-trivial case is if we want to use cut on $\mathbb{T}[\xi] \Rightarrow \psi$, which is a member of Φ, and $\mathbb{U}[\chi] \Rightarrow \xi$. Then ξ is an atom, e.g. $\xi = p$, because Φ is regular. Hence p cannot be the main formula in $\mathbb{U}[\chi] \Rightarrow p$. Therefore the technique for side formulae or the above mentioned cut transformation can be used on it. □

A.2.2 Non-associative Lambek Calculus with only product and join

The variant of cut-elimination for $\mathbf{FNL}(\Phi)$ proved in Theorem A.2.2 has many consequences. Let Φ be a regular set of non-logical axioms and $\mathbb{S} \Rightarrow \varphi$ a sequent such that they contain only connectives from $\{\cdot, \vee\}$. Then $\mathbb{S} \Rightarrow \varphi$ is provable in $\mathbf{FNL}(\Phi)$ iff it is provable using only logical axioms (Id), non-logical axioms from Φ, ($\cdot$L), ($\cdot$R), ($\vee$L), ($\vee$R), and (Cut). We will denote such a calculus $\mathbf{NL}^{\vee}(\Phi)$ to emphasize that the full language of $\mathbf{FNL}(\Phi)$ is not needed. Moreover, $\mathbb{S} \Rightarrow \varphi$ has a proof in $\mathbf{NL}^{\vee}(\Phi)$ iff it has a standard proof in $\mathbf{NL}^{\vee}(\Phi)$.

The choice of $\{\cdot, \vee\}$ is not arbitrary. Our Φ, which will demonstrate the undecidability of the consequence relation, will be in this particular language. Therefore we restrict ourselves to this language from now on. We can also define a natural equivalence relation and normal forms on formulae in this language. Let the following equivalences hold for any formulae φ, ψ, and χ.

$$\begin{aligned}
\varphi &\sim \varphi \vee \varphi && \text{(Idempotency)}\\
\varphi \vee \psi &\sim \psi \vee \varphi && \text{(Commutativity)}\\
\varphi \vee (\psi \vee \chi) &\sim (\varphi \vee \psi) \vee \chi && \text{(Associativity)}\\
\varphi \cdot (\psi \vee \chi) &\sim (\varphi \cdot \psi) \vee (\varphi \cdot \chi) && \text{(Left-Distributivity)}\\
(\varphi \vee \psi) \cdot \chi &\sim (\varphi \cdot \chi) \vee (\psi \cdot \chi) && \text{(Right-Distributivity)}
\end{aligned}$$

Let $\sim^*$ be the reflexive, symmetric, and transitive closure of $\sim$. If we read $\sim$ in the previous equivalences as $\Rightarrow$ and $\Leftarrow$ it is easy to verify that the corresponding sequents are provable in $\mathbf{NL}^{\vee}(\emptyset)$. Hence it is clear that if we prove $\mathbb{S}[\varphi] \Rightarrow \psi$ and $\varphi \sim^* \chi$ then we can also prove $\mathbb{S}[\chi] \Rightarrow \psi$ using (Cut).

Definition A.2.3. A formula φ is *simple* if the only connective occurring in φ is product ($\cdot$).

We define a variant of normal forms for formulae in $\{\cdot, \vee\}$, because any such a formula can be equivalently expressed as a join of simple formulae. It is sufficient to apply left- and right-distributivity from left to right as many times as needed. Moreover, this representation can be considered unique (assuming the idempotency, commutativity, and associativity of join) if we impose an order, e.g. lexicographic, on simple formulae. However, the actual order is not important for us. Therefore we assume that it is fixed in our paper.

Definition A.2.4. A formula ψ is a *simple representation* of a formula φ if $\psi = \bigvee_{i=1}^{n} \chi_i$, all χ_i are simple, and $\varphi \sim^* \psi$. We write $\psi = [\varphi]_s$ and also use notation that $\chi_i \in [\varphi]_s$ if this representation is unique, i.e. n is the minimal possible and simple formulae are ordered.

In order to simplify the notation we will use this representation of formulae. Moreover, we omit products and most parentheses in formulae as we implicitly assume that product is the default connective and parentheses tight to right, e.g. pqr is strictly speaking $(p \cdot (q \cdot r))$.

A.2.3 Tag systems

Tag systems were proposed by Post [Pos43] and they operate on finite words, i.e. finite sequences of letters. Let $\mathcal{A} = \{a_1, \ldots, a_n\}$ be a finite alphabet of letters with no special halting letter. A set of production rules is given by a function $\pi \colon [1, n] \to \mathcal{A}^*$, where $\mathcal{A}^*$ is the set of finite words over $\mathcal{A}$. The computation of the 2-tag system given by $\mathcal{A}$ and π on a word $w \in \mathcal{A}^*$ is defined as follows.

If $|w| < 2$ we terminate. Otherwise, we examine the first letter in w, which is some a_i. Then we delete the first two letters in w and append $\pi(i)$ to the rest of the word after the last letter and obtain a new word.

We repeat this process until it is possible, i.e. we can run forever or at some point we terminate—we obtain a word with less than two letters. We use $\downarrow$ for such a word. In the later case we say that the 2-tag system terminates on w and write $w \rightsquigarrow^*_{\mathcal{A},\pi} \downarrow$. It is well-known, see [CM64], that the halting problem for a 2-tag system, i.e. whether it terminates on a given w, is generally undecidable.

Example A.2.2. Let $\mathcal{A} = \{a_1, a_2\}$, $\pi(1) = a_2$, and $\pi(2) = a_1$ describe a 2-tag system. Then our very elementary system, starting on $w = a_1a_2a_2$, computes as follows:

$$\begin{aligned} w = \; & \underline{a_1}a_2a_2 \\ & \cancel{a_1a_2}\underline{a_2}a_2 \\ & \cancel{a_1a_2a_2a_2}a_1 = \downarrow \end{aligned}$$

A.3 Encoding

In this section we present how to encode a computation of a 2-tag system in the language $\{\cdot, \vee\}$. We have an arbitrary but fixed 2-tag system given by $\mathcal{A} = \{a_1, \ldots, a_n\}$ and $\pi \colon [1, n] \to \mathcal{A}^*$. Assume that $w \in \mathcal{A}^*$ is a word. We recall our simple representation of formulae—a join of formulae containing

only products. For simplicity we also do not write products and most parentheses in formulae as we implicitly assume that product is the default connective and parentheses tight to right.

The encoding is based on a join of simple formulae which read from the left side contain some prefix, letters from $\mathcal{A}$, and end with a symbol (a capital letter) describing their meaning.

Moreover, to simplify the formulation of the encoding the letters from $\mathcal{A}$ are represented in *pairs* starting from the left side. It is convenient as 2-tag systems delete pairs of letters. Hence a pair of letters $a_i a_j$ is represented by c_i^j. When the length of a word is odd then its last letter a_i is simply represented by c_i.

Definition A.3.1. Let $\mathcal{A} = \{a_1, \ldots, a_n\}$ and $\pi\colon [1, n] \to \mathcal{A}^*$ describe a 2-tag system. The set $\mathcal{C}(\mathcal{A})$ of finite words in the *pair notation* over $\mathcal{A}$ is given by $\mathcal{C}(\mathcal{A}) = \mathcal{C}(\mathcal{A})_p^* \cup \mathcal{C}(\mathcal{A})_p^* \times \mathcal{C}(\mathcal{A})_s$, where $\mathcal{C}(\mathcal{A})_p = \{\, c_i^j \mid 1 \leq i, j \leq n \,\}$ and $\mathcal{C}(\mathcal{A})_s = \{c_1, \ldots, c_n\}$. The translation function $\delta\colon \mathcal{A}^* \to \mathcal{C}(\mathcal{A})$ is defined by

$$\delta(w) = \begin{cases} \epsilon & \text{if } w \text{ is the empty word,} \\ c_i & \text{if } w = a_i, \\ c_i^j \delta(v) & \text{if } w = a_i a_j v, \end{cases}$$

where ϵ represents the empty string. We will also use the reverse function $\delta^{-1}\colon \mathcal{C}(\mathcal{A}) \to \mathcal{A}^*$.

Example A.3.1. It holds that $\delta(a_1 a_2 a_1 a_2) = c_1^2 c_1^2$ and $\delta(a_1 a_2 a_1) = c_1^2 c_1$.

Although using our conventions on notation simple formulae may look like sequences, they are trees. Hence the cut rule only enables us to substitute a tree for a subtree in them. As in our representation parentheses tight to right it is easy to process the end of a formula, because it is a subtree. However, when we append or delete letters we have to transfer pieces of information between the beginning and end of words (formulae). To get around this problem, we will use join.

First, we present some basic ideas in Section A.3.1. Second, the needed non-logical axioms are presented in Section A.3.2. However, all the properties of the encoding will be apparent from Sections A.4 and A.5, where the correctness and completeness of this encoding are proved.

A.3.1 The basic ideas behind

As the encoding, given by non-logical axioms, is relatively complex we present the behavior of our encoding first. The rules should be then easier to understand.

State formulae

First, we describe how a state of a 2-tag system is represented by a formula in our encoding. Later on we will call these formulae state formulae. For this purposes the alphabet contains the following symbols:

- $c_1^1, \ldots, c_n^n$ represent the pairs of letters,
- $c_1, \ldots, c_n$ represent the letters,
- e and e' represent the deleted pairs of letters,
- X and X' represent the end of words.

A state of a computation of a 2-tag system is fully described by the word it processes. We check whether its length is at least two, delete the first two letters, and append letters according to π and the first deleted letter. In our encoding we switch deleting and appending. However, it does not matter, because we will append letters only if the length of the processed word is at least two.

We start by representing the word w as $e\delta(w)X$, where the only connective is product and all parentheses tight to right.

Let us look at the 2-tag system from Example A.2.2, i.e. $w = a_1a_2a_2$, $\pi(1) = a_2$, and $\pi(2) = a_1$. Then the initial formula is $ec_1^2c_2X$. The computation continues in such a way that we add $\pi(1)$ and change X to X'. Hence we obtain $ec_1^2c_2^2X'$, where c_2^2 represents a_2a_2, see our pair notation.

The change of the primality of X (and other symbols) will enable us to recognize whether we only added letters or also deleted the corresponding pair of letters. This subtle detail will be essential to the completeness proof. It will ensure that the appending and deleting steps must alternate. Therefore there will be a straightforward translation between state formulae and words occurring in a computation of a 2-tag system.

In $ec_1^2c_2^2X'$ we delete c_1^2, which represents a_1a_2, by changing it to e'. Hence we have $ee'c_2^2X'$. This represents the next state of the computation as both X and the last e are primed. The computation can continue to $ee'c_2^2c_1X$ and by changing c_2^2 to e we get $ee'ec_1X$. However, then the computation cannot proceed as only one letter remains.

From the previous example we can see that during the computation we add letters to the end and change letters at the beginning to e or e'. Therefore correct formulae start with alternating e and e'. If X and the last e are both primed or both not primed then such a formula directly represents a state of a 2-tag system (a word). Otherwise, the letters were

added according to π, but the corresponding first two letters, represented by c_i^j, had not been deleted.

If there are less than two letters from $\mathcal{A}$ and the closest e and X are both primed or both not primed we are in a termination state. As we want an unique representation of termination states we first delete the only letter (if any) from $\mathcal{A}$ and then also e and e' one by one. Therefore all the termination states can be represented by the formula eX.

In our example, we get the following sequence $ee'ec_1X$, $ee'eX$, $ee'X'$, and finally we get eX.

Auxiliary formulae

Nevertheless, it is clear that we must check whether we appended and deleted (changed to e or e') the right letters. Moreover, we must be very particular about the correct alternation of appending and deleting steps. Therefore we have to use some auxiliary formulae, which ensure that the system is simulated faithfully. Hence whenever we execute any such a step we obtain a join of a state formula and an auxiliary formula. This auxiliary formula certifies that the step was used correctly. For this purposes some more symbols are needed:

- A represents the end of general auxiliary formulae,
- C_i and C_i', for $1 \leq i \leq n$, represent the end of auxiliary formulae for adding letters,
- D and D' represent the end of auxiliary formulae for deleting letters,
- d and d' represent the position where a pair of letters was deleted.

The idea is that whenever we obtain an auxiliary formula we must transform it into the general auxiliary formula eA. This ensures that the step was used correctly.

Let us again assume we have $ec_1^2c_2X$. We want to append $\pi(1)$. However, this is possible only if the word represented by this formula begins with a_1 and contains at least one more letter. Hence some c_1^j has to be after the last e. We will be able to prove

$$ec_1^2c_2X \Rightarrow ec_1^2(c_2^2X' \vee c_2C_1),$$

from our non-logical axioms and $ec_1^2(c_2^2X' \vee c_2C_1) \sim^* ec_1^2c_2^2X' \vee ec_1^2c_2C_1$. Here C_1 in $ec_1^2c_2C_1$ says that we appended $\pi(1)$ and hence we want to check whether the word represented by $ec_1^2c_2C_1$ starts with some c_1^j. We can show

that by deleting all the letters but the first pair. Therefore from $ec_1^2c_2C_1$ we obtain $ec_1^2C_1$. Then we get eA, because C_1 matches the first letter in c_1^2, which represents a_1a_2, and the primality of C_1 and e is the same. Moreover, c_1^2 certifies that the original formula contained at least two letters from $\mathcal{A}$ before $\pi(1)$ was appended. It works similarly for C_i', e', and X'.

However, we cannot continue with another appending step, because the primality of X' is different from the primality of e in $ec_1^2c_2^2X'$. Hence we would not be able to transform the auxiliary formula into eA. Therefore we now have to delete the first two letters, c_1^2 in our case. We will be able to prove

$$ec_1^2c_2^2X' \Rightarrow e(e' \vee d')c_2^2X'$$

and $e(e' \vee d')c_2^2X' \sim^* ee'c_2^2X' \vee ed'c_2^2X'$. Here $ed'c_2^2X'$ is not a state formula, because it contains d'. Hence it is an auxiliary formula and we want to obtain eA from it. First, we get $ed'c_2^2D'$. Here D' says that we deleted the first two letters, which are represented by d'. The procedure is then similar to the previous case, we delete all the letters between d' and D'. From $ed'c_2^2D'$ we obtain $ed'D'$ and then eA. This last step is possible for the following two reasons. First, D' and d' are primed and e is not primed. Second, there is no further symbol between e and d'. It works similarly for X, but we use D, d and get e in the state formula.

Note that we could now continue by appending $\pi(2)$, because e' and X' in $ee'c_2^2X'$ are both primed, but not by deleting c_2^2, because we would not be able to transform the obtained auxiliary formula into eA.

A.3.2 Actual representation

From the previous text it should be clear that the question whether a 2-tag system given by $\mathcal{A}$ and π terminates on a $w \in \mathcal{A}^*$ will be translated into the question whether

$$e\delta(w)X \Rightarrow eX \vee eA$$

is provable in $\mathbf{NL}^\vee(\Phi[\mathcal{A}, \pi])$, where $\Phi[\mathcal{A}, \pi]$ is a finite set of non-logical axioms given in the following definition.

Definition A.3.2. Let $\mathcal{A} = \{a_1, \ldots, a_n\}$ and $\pi\colon [1, n] \to \mathcal{A}^*$ describe a 2-tag system. The set of non-logical axioms $\Phi[\mathcal{A}, \pi]$ contains

$$e'eX \Rightarrow e'X' \qquad \text{(A.1)} \qquad\qquad ee'X' \Rightarrow eX \qquad \text{(A.4)}$$
$$ec_iX \Rightarrow eX \qquad \text{(A.2)} \qquad\qquad e'c_iX' \Rightarrow e'X' \qquad \text{(A.5)}$$
$$ee'A \Rightarrow eA \qquad \text{(A.3)} \qquad\qquad e'eA \Rightarrow e'A \qquad \text{(A.6)}$$

$$ec_i^j C_i \Rightarrow eA \quad \text{(A.7)} \qquad e'c_i^j C'_i \Rightarrow e'A \quad \text{(A.12)}$$

$$c_j^k c_l C_i \Rightarrow c_j^k C_i \quad \text{(A.8)} \qquad c_j^k c_l C'_i \Rightarrow c_j^k C'_i \quad \text{(A.13)}$$

$$c_j^k c_l^m C_i \Rightarrow c_j^k C_i \quad \text{(A.9)} \qquad c_j^k c_l^m C'_i \Rightarrow c_j^k C'_i \quad \text{(A.14)}$$

$$c_j X \Rightarrow \delta(a_j\, \pi(i))X' \vee c_j C_i \quad \text{(A.10)} \qquad c_j X' \Rightarrow \delta(a_j\, \pi(i))X \vee c_j C'_i \quad \text{(A.15)}$$

$$c_j^k X \Rightarrow c_j^k \delta(\pi(i))X' \vee c_j^k C_i \quad \text{(A.11)} \qquad c_j^k X' \Rightarrow c_j^k \delta(\pi(i))X \vee c_j^k C'_i \quad \text{(A.16)}$$

$$ed'D' \Rightarrow eA \quad \text{(A.17)} \qquad e'dD \Rightarrow e'A \quad \text{(A.24)}$$

$$d'c_i D' \Rightarrow d'D' \quad \text{(A.18)} \qquad dc_i D \Rightarrow dD \quad \text{(A.25)}$$

$$d'c_i^j D' \Rightarrow d'D' \quad \text{(A.19)} \qquad dc_i^j D \Rightarrow dD \quad \text{(A.26)}$$

$$c_i^j c_k D' \Rightarrow c_i^j D' \quad \text{(A.20)} \qquad c_i^j c_k D \Rightarrow c_i^j D \quad \text{(A.27)}$$

$$c_i^j c_k^l D' \Rightarrow c_i^j D' \quad \text{(A.21)} \qquad c_i^j c_k^l D \Rightarrow c_i^j D \quad \text{(A.28)}$$

$$X' \Rightarrow D' \quad \text{(A.22)} \qquad X \Rightarrow D \quad \text{(A.29)}$$

$$c_i^j \Rightarrow e' \vee d' \quad \text{(A.23)} \qquad c_i^j \Rightarrow e \vee d \quad \text{(A.30)}$$

for every $i, j, k, l, m \in [1, n]$.

Remark. The set $\Phi[\mathcal{A}, \pi]$ is not regular, but we can replace all the products on the left side of sequents by commas and obtain regular sequents, i.e. we use σ. Lemma A.2.1 says that this does not change provability. Therefore we can always assume that $\Phi[\mathcal{A}, \pi]$ is regular.

The intuitive and simplified meaning of these non-logical axioms is following. By (A.1,A.4) and (A.3,A.6) we say that e and e' have to alternate. The axioms (A.1,A.4) and (A.2,A.5) say that a terminating word can contain no or one letter from $\mathcal{A}$ and that X and the last e are simultaneously primed or not primed.

It is important to notice that some rules have two variants, because we use the pair notation and words can have odd or even length.

The axioms (A.7–A.9) (or A.12–A.14) describe how formulae containing C_i (or C'_i) have to look like, i.e. c_i^j is right after e (or e'). Likewise, the axioms (A.17–A.21) (or A.24–A.28) describe formulae containing D' (or D), i.e. d' (or d) is right after e (or e').

The meaning of (A.10,A.11) (or A.15,A.16) is more complicated. If we apply them to a simple formula we obtain a join of two simple formulae, using the simple representation. The first of them contains appended $\pi(i)$ and changed primality of X. The second formula contains C_i (or C'_i) and

we want to obtain eA from this formula. This is possible if c_i^j is right after e (or e') and it certifies that the rule was used correctly.

If the axioms (A.23) (or A.30) are applied to a simple formula we again obtain a join of two simple formulae, using the simple representation. The first one contains e' (or e) instead of c_i^j. The second one contains d' (or d) instead of it. We want to eliminate, obtain eA, the second formula. The axioms (A.22) (or A.29) allow us to rewrite such a formula to one ending with D' (or D). We can eliminate it, obtain eA, if d' (or d) is right after e (or e').

A.4 Correctness of encoding

We presented the set of non-logical axioms $\Phi[\mathcal{A}, \pi]$ for a given 2-tag system described by $\mathcal{A} = \{a_1, \ldots, a_n\}$ and $\pi\colon [1, n] \to \mathcal{A}^*$. From now on, this 2-tag system will be fixed.

In this section we show that the representation is correct, i.e. what is computed by the 2-tag system can be also proved using $\Phi[\mathcal{A}, \pi]$ and our encoding. As we already indicated we divide formulae into two groups: formulae representing a state of the 2-tag system and auxiliary formulae.

In the following text we use $\langle ee' \rangle^m$ to describe repeating e and e'. Let $\langle ee' \rangle^0$ be the empty sequence and $\langle ee' \rangle^{m+1} = ee' \langle ee' \rangle^m$. Moreover, to simplify our presentation we handle simple formulae as sequences. For our purposes here, we can do it, but we have to be aware of their real representation, i.e. they are trees containing products and parentheses which tight to right. Therefore if we assume that $evX = ec_1^2c_2X$ then the subsequence $v = c_1^2c_2$ is not here even a (sub)formula, because $ec_1^2c_2X$ is strictly speaking $(e \cdot (c_1^2 \cdot (c_2 \cdot X)))$. However, we can say that $v \in \mathcal{C}(\mathcal{A})$ and hence even $\delta^{-1}(v) \in \mathcal{A}^*$, because it represents the word $a_1a_2a_2$.

Definition A.4.1. A formula φ is an *auxiliary formula* in $\mathcal{A}$ iff it is one of the following

1. $\langle ee' \rangle^m eA$,
2. $\langle ee' \rangle^m ec_i^j vC_i$,
3. $\langle ee' \rangle^m ed'vD'$,
4. $\langle ee' \rangle^m ed'vX'$,
5. $\langle ee' \rangle^m ee'A$,
6. $\langle ee' \rangle^m ee'c_i^j vC_i'$,
7. $\langle ee' \rangle^m ee'dvD$,
8. $\langle ee' \rangle^m ee'dvX$,

for $v \in \mathcal{C}(\mathcal{A})$, $m \geq 0$, and $1 \leq i, j \leq n$. We define the set of all auxiliary formulae in $\mathcal{A}$ by $\Gamma_{\mathcal{A}} = \{\, \psi \mid \psi \text{ is an auxiliary formula in } \mathcal{A} \,\}$.

In the following lemmata we simplify proofs using Lemma A.2.1. If we want to use (Cut) with the cut formula φ, then there has to be a structure $\mathbb{S}[\varphi]$. However, sometimes we want to "substitute" for a subformula φ in a formula χ. This is possible if $\chi = \rho(\mathbb{S}[\varphi])$, because $\mathbb{S}[\varphi] \Rightarrow \psi$ is provable iff $\chi \Rightarrow \psi$ is provable by Lemma A.2.1. Then using (Cut) on $\mathbb{S}[\varphi] \Rightarrow \psi$ and $\mathbb{T} \Rightarrow \varphi$ we obtain $\mathbb{S}[\mathbb{T}] \Rightarrow \psi$. Finally, using again Lemma A.2.1 we get $\rho(\mathbb{S}[\mathbb{T}]) \Rightarrow \psi$. We denote such an application of (Cut) by (Cut*). Similarly, we can use ($\vee$L) and denote it ($\vee$L*).

Lemma A.4.1. *If $\varphi \in \Gamma_{\mathcal{A}}$ then $\varphi \Rightarrow eA$ is provable in $\mathbf{NL}^\vee(\Phi[\mathcal{A}, \pi])$.*

Proof. We check all the cases. If $\varphi = \langle ee' \rangle^m eA$, for $m > 0$, then we use (A.6) and (A.3) from $\Phi[\mathcal{A}, \pi]$

$$\dfrac{\dfrac{\dfrac{}{\vdots}\ (\text{Cut}^\star)\ \dfrac{ee'A \Rightarrow eA \qquad e'eA \Rightarrow e'A}{\langle ee' \rangle^1 eA \Rightarrow eA} \qquad ee'A \Rightarrow eA}{\langle ee' \rangle^2 A \Rightarrow eA}}{\vdots}\ (\text{Cut}^\star)\ \dfrac{\langle ee' \rangle^m A \Rightarrow eA \qquad e'eA \Rightarrow e'A}{\langle ee' \rangle^m eA \Rightarrow eA}$$

We get $\langle ee' \rangle^m ee'A \Rightarrow eA$ by one more application of (A.3).

If $\varphi = \langle ee' \rangle^m ec_i^j vC_i$ we first obtain

$$\dfrac{\dfrac{}{\vdots}\ (\text{Cut}^\star)\ \dfrac{\langle ee' \rangle^m eA \Rightarrow eA \qquad ec_i^j C_i \Rightarrow eA}{\langle ee' \rangle^m ec_i^j C_i \Rightarrow eA}}{\vdots}$$

using (A.7) and then use (A.9) as many times as needed, i.e. $\lfloor |\delta^{-1}(v)|/2 \rfloor$-times. If $\delta^{-1}(v)$ has odd length then we have to use also (A.8). The situation with $\langle ee' \rangle^m ee' c_i^j vC_i'$ is completely analogous.

If we want to prove $\langle ee' \rangle^m ed'vD' \Rightarrow eA$ (or $\langle ee' \rangle^m ee'dvD \Rightarrow eA$) we start from $\langle ee' \rangle^m eA \Rightarrow eA$ (or $\langle ee' \rangle^m ee'A \Rightarrow eA$) and use (A.17–A.21) (or A.24–A.28) in a very similar fashion. Finally, using (A.22) (or A.29) on $\langle ee' \rangle^m ed'vD' \Rightarrow eA$ (or $\langle ee' \rangle^m ee'dvD \Rightarrow eA$) we immediately get $\langle ee' \rangle^m ed'vX' \Rightarrow eA$ (or $\langle ee' \rangle^m ee'dvX \Rightarrow eA$). □

Corollary A.4.2. *If $\varphi \in \Gamma_{\mathcal{A}}$ then $\varphi \Rightarrow eX \vee eA$ is provable in $\mathbf{NL}^\vee(\Phi[\mathcal{A}, \pi])$.*

Definition A.4.2. A formula φ is a *state formula* in $\mathcal{A}$ iff it is one of the following

1. $\langle ee' \rangle^m evX$,
2. $\langle ee' \rangle^m ee' c_i^j vX$,
3. $\langle ee' \rangle^m ee' vX'$,
4. $\langle ee' \rangle^m ec_i^j vX'$,

for $v \in \mathcal{C}(\mathcal{A})$, $m \geq 0$, and $1 \leq i, j \leq n$. We define the set of all state formulae in $\mathcal{A}$ by $\Lambda_{\mathcal{A}} = \{ \psi \mid \psi \text{ is a state formula in } \mathcal{A} \}$.

The function τ which translates a state formula φ into a word in $\mathcal{A}^*$ is defined by

$$\tau(\varphi) = \delta^{-1}(v).$$

Example A.4.1. It holds that $\tau(ec_1^2 c_2 X) = a_1 a_2 a_2$, $\tau(ec_1^2 c_2^2 X') = a_2 a_2$, and $\tau(ee' c_1 X') = a_1 = \downarrow$.

Lemma A.4.3. *If* $w \rightsquigarrow^*_{\mathcal{A},\pi} \downarrow$ *then* $\langle ee' \rangle^m e\delta(w)X \Rightarrow eX \vee eA$ *and* $\langle ee' \rangle^m ee'\delta(w)X' \Rightarrow eX \vee eA$ *are provable in* $\mathbf{NL}^{\vee}(\Phi[\mathcal{A}, \pi])$ *for any* $m \geq 0$.

Proof. We prove this by induction on the length of the 2-tag derivation. If w is in a termination state it means w is the empty sequence or $w = a_i$, for some $i \in [1, n]$. We use (A.4) and (A.1) m-times to obtain $\langle ee' \rangle^m eX \Rightarrow eX$. Using (A.2) we get $\langle ee' \rangle^m ec_i X \Rightarrow eX$, for $1 \leq i \leq n$. One more application of (A.4) on $\langle ee' \rangle^m eX \Rightarrow eX$ leads to $\langle ee' \rangle^m ee' X' \Rightarrow eX$. Then using (A.5) we get $\langle ee' \rangle^m ee' c_i X' \Rightarrow eX$, for $1 \leq i \leq n$. We complete all these cases by $(\vee\mathrm{R})$.

Let us assume that one step computation of our 2-tag system leads from w to v and $\langle ee' \rangle^m ee'\delta(v)X' \Rightarrow eX \vee eA$ and $\langle ee' \rangle^m ee' e\delta(v)X \Rightarrow eX \vee eA$ are provable in $\mathbf{NL}^{\vee}(\Phi[\mathcal{A}, \pi])$. It means that there are indexes i and j such that $w = a_i a_j u$ and $v = u\,\pi(i)$. Using Corollary A.4.2 and then (A.23) we obtain

$$(\vee\mathrm{L}^{\star})\,\frac{\begin{array}{c}\vdots\\ \langle ee' \rangle^m ee'\delta(u\,\pi(i))X' \Rightarrow eX \vee eA\end{array} \quad \begin{array}{c}\vdots\\ \langle ee' \rangle^m ed'\delta(u\,\pi(i))X' \Rightarrow eX \vee eA\end{array}}{(\mathrm{Cut}^{\star})\,\dfrac{\langle ee' \rangle^m e(e' \vee d')\delta(u\,\pi(i))X' \Rightarrow eX \vee eA \qquad c_i^j \Rightarrow e' \vee d'}{\langle ee' \rangle^m ec_i^j \delta(u\,\pi(i))X' \Rightarrow eX \vee eA}}$$

If u has odd length, $u = ta_k$, then $\delta(ta_k\,\pi(i)) = \delta(t)\delta(a_k\,\pi(i))$, and using Corollary A.4.2 and then (A.10) we get

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \langle ee' \rangle^m ec_i^j \delta(t)\delta(a_k\,\pi(i))X' \Rightarrow eX \vee eA\end{array} \quad \begin{array}{c}\vdots\\ \langle ee' \rangle^m ec_i^j \delta(t)c_k C_i \Rightarrow eX \vee eA\end{array}}{(\mathrm{Cut}^{\star})\,\dfrac{\langle ee' \rangle^m ec_i^j \delta(t)(\delta(a_k\,\pi(i))X' \vee c_k C_i) \Rightarrow eX \vee eA \qquad c_k X \Rightarrow \delta(a_k\,\pi(i))X' \vee c_k C_i}{\langle ee' \rangle^m ec_i^j \delta(t)c_k X \Rightarrow eX \vee eA}}(\vee\mathrm{L}^{\star})$$

where $c_i^j\delta(t)c_k = \delta(a_ia_ju) = \delta(w)$. If u has even length we use (A.11).

The proof of $\langle ee'\rangle^m ee'c_i^j\delta(u)X' \Rightarrow eX \vee eA$ using the provability of $\langle ee'\rangle^m ee'e\delta(v)X \Rightarrow eX \vee eA$ is completely analogous. □

Corollary A.4.4. *If* $w \rightsquigarrow^*_{\mathcal{A},\pi} \downarrow$ *then* $e\delta(w)X \Rightarrow eX \vee eA$ *is provable in* $\mathbf{NL}^\vee(\Phi[\mathcal{A},\pi])$.

A.5 Completeness of encoding

We have proved that our encoding can simulate any terminating computation of our fixed 2-tag system given by $\mathcal{A} = \{a_1, \ldots, a_n\}$ and $\pi\colon [1,n] \to \mathcal{A}^*$. It remains to prove the other direction—any proof of a sequent expressing that the system terminates can be translated into a terminating computation of the given 2-tag system.

Recall the definition of our simple representation, see Definition A.2.4, the translation ρ, which replaces all commas in structures by products, and the reverse translation σ, see Definition A.2.2.

As was already mentioned we can assume that all the members of $\Phi[\mathcal{A},\pi]$ are regular, i.e. any $\alpha \Rightarrow \beta$ from $\Phi[\mathcal{A},\pi]$ is treated as $\sigma(\alpha) \Rightarrow \beta$, see Lemma A.2.1. We know from Theorem A.2.2 that $\mathbb{S} \Rightarrow \varphi$ is provable in $\mathbf{NL}^\vee(\Phi[\mathcal{A},\pi])$ iff it has a standard proof in $\mathbf{NL}^\vee(\Phi[\mathcal{A},\pi])$. Therefore the following three lemmata are proved by induction on the height of the proof, which is the length of the longest branch in its tree representation, with only principal cuts.

We want to prove that if $\mathbb{S} \Rightarrow eX \vee eA$ is provable then the simple representation of $\rho(\mathbb{S})$ contains only auxiliary formulae or state formulae that represent words with terminating computations.

The main reason, why we use the equivalence $\sim^*$ on formulae and simple representation, is the rule ($\vee$L). This rule enables us to join two structures, e.g. from $ec_1X \Rightarrow eX \vee eA$ and $ec_2X \Rightarrow eX \vee eA$ we can easily prove $e(c_1 \vee c_2)X \Rightarrow eX \vee eA$ and $(e \vee e)c_1X \Rightarrow eX \vee eA$. Our simple representation is a way how to handle similar obstacles.

Lemma A.5.1. *If* $\mathbb{S} \Rightarrow eA$ *is provable in* $\mathbf{NL}^\vee(\Phi[\mathcal{A},\pi])$ *then for all* $\psi \in [\rho(\mathbb{S})]_s$ *hold* $\psi \in \Gamma_{\mathcal{A}}$.

Proof. The proof is by induction on the height of the standard proof. If the height is zero then the only possibilities are (Id) or members of $\Phi[\mathcal{A},\pi]$. Hence $[\rho(\mathbb{S})]_s$ can contain only eA, $ee'A$, $ed'D'$, or $ec_i^jC_i$, for any $i,j \in [1,n]$. All these formulae are elements of $\Gamma_{\mathcal{A}}$.

We assume that this lemma holds for all standard proofs with height at most n. Fix an arbitrary proof of $\mathbb{S} \Rightarrow eA$ with height $n+1$. We have to check all the possible last steps.

It is impossible that $(\vee\mathrm{R})$ is the last step and if it is $(\cdot\mathrm{L})$ then we get $\mathbb{S} \Rightarrow eA$ from some $\mathbb{S}' \Rightarrow eA$, where $\rho(\mathbb{S}) = \rho(\mathbb{S}')$.

If $(\vee\mathrm{L})$ is the last step then it means that there are $\mathbb{W}$, χ, and ξ such that $\mathbb{S} = \mathbb{W}[\chi \vee \xi]$, $\mathbb{W}[\chi] \Rightarrow eA$, and $\mathbb{W}[\xi] \Rightarrow eA$. Hence $\psi \in [\rho(\mathbb{S})]_s$ iff $\psi \in [\rho(\mathbb{W}[\chi])]_s$ or $\psi \in [\rho(\mathbb{W}[\xi])]_s$.

If $(\cdot\mathrm{R})$ is the last step then there exist $\mathbb{U}$ and $\mathbb{V}$ such that $\mathbb{S} = (\mathbb{U}, \mathbb{V})$, $\mathbb{U} \Rightarrow e$, and $\mathbb{V} \Rightarrow A$. It is easy to prove by induction that $e = [\mathbb{U}]_s$ and $A = [\mathbb{V}]_s$. Hence $eA = [(\mathbb{U}, \mathbb{V})]_s$.

The last case is when we use a principal cut. Let us assume that from $\mathbb{S}' \Rightarrow eA$ and a member of $\Phi[\mathcal{A}, \pi]$ we obtain $\mathbb{S} \Rightarrow eA$.

If an axiom $\sigma(\alpha) \Rightarrow \beta$ with no join is used then it is sufficient to show for any $\rho(\mathbb{T}[\beta]) \in [\rho(\mathbb{S}')]_s$, i.e. $\rho(\mathbb{T}[\beta]) \in \Gamma_{\mathcal{A}}$, that $\rho(\mathbb{T}[\alpha]) \in \Gamma_{\mathcal{A}}$. We check all the possible axioms and assume $\mathbb{T}[\beta]$ is an arbitrary but fixed.

If we use one of (A.3), (A.6–A.9), (A.12–A.14), (A.17–A.21), or (A.24–A.28) then from $\mathbb{T}[\beta]$, where $\rho(\mathbb{T}[\beta]) \in \Gamma_{\mathcal{A}}$, we obtain $\mathbb{T}[\sigma(\alpha)]$ such that $\rho(\mathbb{T}[\alpha]) \in \Gamma_{\mathcal{A}}$.

If we use (A.22) or (A.29) then $\rho(\mathbb{T}[\beta])$, which is in $\Gamma_{\mathcal{A}}$, is equal to some $\langle ee' \rangle^m ed'vD'$ or $\langle ee' \rangle^m ee'dvD$. Therefore the formula $\rho(\mathbb{T}[\alpha])$ is also in $\Gamma_{\mathcal{A}}$.

The other axioms from $\Phi[\mathcal{A}, \pi]$ cannot be applied, because the induction hypothesis would be violated as we will show in the rest of the proof.

If (A.1), (A.2), (A.4), or (A.5) were used it would mean that $\mathbb{T}[\beta]$ contains eX or $e'X'$. Hence $\rho(\mathbb{T}[\beta]) \notin \Gamma_{\mathcal{A}}$.

We have to also check axioms $\sigma(\alpha) \Rightarrow \beta \vee \gamma$ containing join. It suffices to show for any two formulae $\rho(\mathbb{T}[\beta]) \in [\rho(\mathbb{S}')]_s$ and $\rho(\mathbb{T}[\gamma]) \in [\rho(\mathbb{S}')]_s$, i.e. $\rho(\mathbb{T}[\beta]), \rho(\mathbb{T}[\gamma]) \in \Gamma_{\mathcal{A}}$, that $\rho(\mathbb{T}[\alpha]) \in \Gamma_{\mathcal{A}}$, because $(\rho(\mathbb{T}[\beta]) \vee \rho(\mathbb{T}[\gamma])) \sim^* \rho(\mathbb{T}[\beta \vee \gamma])$. Let $\mathbb{T}[\beta]$ and $\mathbb{T}[\gamma]$ be arbitrary but fixed.

If (A.10) or (A.11) were used then we could clearly assume $\rho(\mathbb{T}[\beta])$ ends with X' and $\rho(\mathbb{T}[\gamma])$ with C_i. Moreover, they are members of $\Gamma_{\mathcal{A}}$. Therefore $\rho(\mathbb{T}[\gamma])$ has to be equal to some $\langle ee' \rangle^m ec_i^j vC_i$ and hence $\rho(\mathbb{T}[\beta]) = \langle ee' \rangle^m ec_i^j \delta(\delta^{-1}(v)\, \pi(i))X'$, but then $\rho(\mathbb{T}[\beta]) \notin \Gamma_{\mathcal{A}}$. It follows that (A.10) and (A.11) are not applicable in this case. Similarly for (A.15) and (A.16).

If (A.23) were used then we could clearly assume $\rho(\mathbb{T}[\gamma])$ contains d'. It is impossible that $\rho(\mathbb{T}[\gamma]) = \langle ee' \rangle^m ed'vD'$ and $\rho(\mathbb{T}[\beta]) = \langle ee' \rangle^m ee'vD'$ as $\rho(\mathbb{T}[\beta]) \notin \Gamma_{\mathcal{A}}$. Therefore $\rho(\mathbb{T}[\gamma]) = \langle ee' \rangle^m ed'vX'$ and $\rho(\mathbb{T}[\beta]) = \langle ee' \rangle^m ee'vX'$, but then also $\rho(\mathbb{T}[\beta]) \notin \Gamma_{\mathcal{A}}$. Similarly for (A.30). □

Lemma A.5.2. *If $\mathbb{S} \Rightarrow eX$ is provable in $\mathbf{NL}^{\vee}(\Phi[\mathcal{A}, \pi])$ then for all $\psi \in [\rho(\mathbb{S})]_s$ hold $\psi \in \Lambda_{\mathcal{A}}$ and $\tau(\psi) = \downarrow$.*

Proof. The proof is by induction on the height of the standard proof. If the height is zero then the only possibilities are (Id) or members of $\Phi[\mathcal{A}, \pi]$. Hence $[\rho(\mathbb{S})]_s$ can contain only eX, $ee'X'$, or ec_iX, for any $i \in [1, n]$. All these formulae are trivially equal to $\downarrow$ under τ.

We assume that this lemma holds for all standard proofs with height at most n. Fix an arbitrary proof of $\mathbb{S} \Rightarrow eA$ with height $n + 1$. We need to consider all the possible last steps.

It is impossible that ($\vee$R) is the last step and if it is ($\cdot$L) or ($\vee$L) then we can use arguments from the previous lemma.

If ($\cdot$R) is the last step then $\mathbb{S} = (\mathbb{U}, \mathbb{V})$, $\mathbb{U} \Rightarrow e$, and $\mathbb{V} \Rightarrow X$. It is easy to prove by induction that $e = [\mathbb{U}]_s$ and $X = [\mathbb{V}]_s$. Hence $eX = [(\mathbb{U}, \mathbb{V})]_s$.

The last case is when we use a principal cut. Let us assume that from $\mathbb{S}' \Rightarrow eA$ and a member of $\Phi[\mathcal{A}, \pi]$ we obtain $\mathbb{S} \Rightarrow eA$. It is clear that only the axioms (A.1), (A.2), (A.4), and (A.5) from $\Phi[\mathcal{A}, \pi]$ are applicable. The other axioms contain on the right side a symbol, which does not occur in $\Lambda_{\mathcal{A}}$. This violates the induction hypothesis.

As an axiom $\sigma(\alpha) \Rightarrow \beta$ with no join is used, it is sufficient to show for any $\rho(\mathbb{T}[\beta]) \in [\rho(\mathbb{S}')]_s$, i.e. $\tau(\rho(\mathbb{T}[\beta])) = \downarrow$, that $\tau(\rho(\mathbb{T}[\alpha])) = \downarrow$. Fix $\mathbb{T}[\beta]$ and we know that the axiom (A.1), (A.2), (A.4), or (A.5) is used. Hence $\mathbb{T}[\beta]$ contains eX or $e'X'$. We conclude from $\rho(\mathbb{T}[\beta]) \in \Lambda_{\mathcal{A}}$ that $\tau(\rho(\mathbb{T}[\alpha])) = \downarrow$. □

Lemma A.5.3. *If $\mathbb{S} \Rightarrow eX \vee eA$ is provable in $\mathbf{NL}^{\vee}(\Phi[\mathcal{A}, \pi])$ then for all $\psi \in [\rho(\mathbb{S})]_s$ hold*

1. *$\psi \in \Gamma_{\mathcal{A}}$, or*
2. *$\psi \in \Lambda_{\mathcal{A}}$ and $\tau(\psi) \rightsquigarrow^*_{\mathcal{A},\pi} \downarrow$.*

Proof. The proof is by induction on the height of the standard proof. If the height is zero then the only possibility is (Id). Hence $[\rho(\mathbb{S})]_s$ contains eX and eA. We know that $\tau(eX) = \downarrow$ and $eA \in \Gamma_{\mathcal{A}}$. Note that τ is defined only for state formulae and therefore the application of τ on any formula implies that this formula is in $\Lambda_{\mathcal{A}}$.

We assume that this lemma holds for all standard proofs with height at most n. Fix an arbitrary proof of $\mathbb{S} \Rightarrow eX \vee eA$ with height $n + 1$. We need to consider all the possible last steps.

It is impossible that ($\cdot$R) is the last step and if it is ($\cdot$L) or ($\vee$L) then we can use arguments from Lemma A.5.1.

If ($\vee$R) is the last step then there are two possibilities. If we use ($\vee$R) on $\mathbb{S} \Rightarrow eA$ then for all $\psi \in [\rho(\mathbb{S})]_s$ hold $\psi \in \Gamma_{\mathcal{A}}$ by Lemma A.5.1. And if we use ($\vee$R) on $\mathbb{S} \Rightarrow eX$ then for all $\psi \in [\rho(\mathbb{S})]_s$ hold $\tau(\psi) = \downarrow$ by Lemma A.5.2.

The last case is when we use a principal cut. Let us assume that from $\mathbb{S}' \Rightarrow eX \vee eA$ and a member of $\Phi[\mathcal{A}, \pi]$ we obtain $\mathbb{S} \Rightarrow eX \vee eA$.

If an axiom $\sigma(\alpha) \Rightarrow \beta$ with no join is used then it is sufficient to show for any $\rho(\mathbb{T}[\beta]) \in [\rho(\mathbb{S}')]_s$, i.e. $\rho(\mathbb{T}[\beta]) \in \Gamma_{\mathcal{A}} \cup \Lambda_{\mathcal{A}}$, that $\rho(\mathbb{T}[\alpha]) \in \Gamma_{\mathcal{A}}$ or $\tau(\rho(\mathbb{T}[\alpha])) \rightsquigarrow^*_{\mathcal{A},\pi} \downarrow$. We have to check all the possible axioms and assume $\mathbb{T}[\beta]$ is an arbitrary but fixed.

If we use the axiom (A.1), (A.2), (A.4), or (A.5) it means that $\mathbb{T}[\beta]$ contains eX or $e'X'$. Hence $\rho(\mathbb{T}[\beta]) \in \Lambda_{\mathcal{A}}$ and $\tau(\rho(\mathbb{T}[\alpha])) = \downarrow$.

If we use one of (A.3), (A.6–A.9), (A.12–A.14), (A.17–A.21), or (A.24–A.28) then $\rho(\mathbb{T}[\beta]) \in \Gamma_{\mathcal{A}}$. Applying any of these axioms we obtain $\mathbb{T}[\sigma(\alpha)]$ such that $\rho(\mathbb{T}[\alpha]) \in \Gamma_{\mathcal{A}}$.

If we use (A.22) or (A.29) then $\rho(\mathbb{T}[\beta]) \in \Gamma_{\mathcal{A}}$, because $\rho(\mathbb{T}[\beta])$ is equal to some $\langle ee' \rangle^m ed'vD'$ or $\langle ee' \rangle^m ee'dvD$ and hence $\rho(\mathbb{T}[\alpha]) \in \Gamma_{\mathcal{A}}$.

We have to also check axioms $\sigma(\alpha) \Rightarrow \beta \vee \gamma$ containing join. In this case two formulae $\rho(\mathbb{T}[\beta]) \in [\rho(\mathbb{S}')]_s$ and $\rho(\mathbb{T}[\gamma]) \in [\rho(\mathbb{S}')]_s$, i.e. $\rho(\mathbb{T}[\beta]), \rho(\mathbb{T}[\gamma]) \in \Gamma_{\mathcal{A}} \cup \Lambda_{\mathcal{A}}$, are sufficient as $(\rho(\mathbb{T}[\beta]) \vee \rho(\mathbb{T}[\gamma])) \sim^* \rho(\mathbb{T}[\beta \vee \gamma])$. We have to show that $\rho(\mathbb{T}[\alpha]) \in \Gamma_{\mathcal{A}}$ or $\tau(\rho(\mathbb{T}[\alpha])) \rightsquigarrow^*_{\mathcal{A},\pi} \downarrow$. Let $\mathbb{T}[\beta]$ and $\mathbb{T}[\gamma]$ be arbitrary but fixed.

If we use (A.10) or (A.11) then we can clearly assume $\rho(\mathbb{T}[\beta])$ ends with X' and $\rho(\mathbb{T}[\gamma])$ with C_i. Therefore $\rho(\mathbb{T}[\gamma])$ is equal to some $\langle ee' \rangle^m ec_i^j vC_i$ and hence $\rho(\mathbb{T}[\beta]) = \langle ee' \rangle^m ec_i^j \delta(\delta^{-1}(v)\, \pi(i))X'$. Hence $\rho(\mathbb{T}[\alpha]) = \langle ee' \rangle^m ec_i^j vX$ and $\tau(\rho(\mathbb{T}[\alpha])) = a_i a_j \delta^{-1}(v)$. We know that $\tau(\rho(\mathbb{T}[\beta])) = \delta^{-1}(v)\, \pi(i) \rightsquigarrow^*_{\mathcal{A},\pi} \downarrow$ and hence $\tau(\rho(\mathbb{T}[\alpha])) \rightsquigarrow^*_{\mathcal{A},\pi} \downarrow$, because we get $\delta^{-1}(v)\, \pi(i)$ from $a_i a_j \delta^{-1}(v)$ by a single step of the 2-tag system. Similarly for (A.15) and (A.16).

If we use (A.23) then we can clearly assume $\rho(\mathbb{T}[\gamma])$ contains d'. It is impossible that $\rho(\mathbb{T}[\gamma]) = \langle ee' \rangle^m ed'vD'$ and $\rho(\mathbb{T}[\beta]) = \langle ee' \rangle^m ee'vD'$, because $\rho(\mathbb{T}[\beta]) \notin \Gamma_{\mathcal{A}} \cup \Lambda_{\mathcal{A}}$. Therefore $\rho(\mathbb{T}[\gamma]) = \langle ee' \rangle^m ed'vX'$ and $\rho(\mathbb{T}[\beta]) = \langle ee' \rangle^m ee'vX'$. Hence $\rho(\mathbb{T}[\alpha]) = \langle ee' \rangle^m ec_i^j vX'$ and $\tau(\rho(\mathbb{T}[\alpha])) = \delta^{-1}(v) = \tau(\rho(\mathbb{T}[\beta])) \rightsquigarrow^*_{\mathcal{A},\pi} \downarrow$. Similarly for (A.30). □

Corollary A.5.4. *If $e\delta(w)X \Rightarrow eX \vee eA$ is provable in $\mathbf{NL}^\vee(\Phi[\mathcal{A}, \pi])$ then $w \rightsquigarrow^*_{\mathcal{A},\pi} \downarrow$.*

As the halting problem for 2-tag systems is generally undecidable Corollaries A.4.4 and A.5.4 give us that the consequence relation in $\mathbf{NL}^\vee$, and therefore in **FNL**, is undecidable.

A.6 Some possible modifications

In this section we discuss some easy modifications of our result. Although they could influence the validity of the completeness of encoding proved in

the previous section, we will show that they have no or negligible effect on the proofs of Lemmata A.5.1–A.5.3.

A.6.1 Adding structural rules

We can add structural rules to our non-associative calculus, cf. [GO10]. The rules of *exchange* (e), *contraction* (c), and *left-weakening* or *integrality* (i) are defined as:

$$\text{(e)}\ \frac{\mathbb{S}[(\mathbb{T},\mathbb{U})]\Rightarrow\psi}{\mathbb{S}[(\mathbb{U},\mathbb{T})]\Rightarrow\psi} \qquad \text{(c)}\ \frac{\mathbb{S}[(\mathbb{T},\mathbb{T})]\Rightarrow\psi}{\mathbb{S}[\mathbb{T}]\Rightarrow\psi} \qquad \text{(i)}\ \frac{\mathbb{S}[\varepsilon]\Rightarrow\psi}{\mathbb{S}[\mathbb{T}]\Rightarrow\psi}$$

It is obvious that our construction fails with the rule of left-weakening. In fact then the consequence relation is decidable, see [BA05].

However, our construction works with exchange and contraction. It is easy to check that Theorem A.2.2 is provable when the rule of exchange is added. If the rule of contraction is added then the only non-trivial case is when

$$\text{(c)}\ \frac{\begin{array}{c}\vdots\\ \mathbb{T}[(\mathbb{U}[\chi],\mathbb{U}[\chi])]\Rightarrow\psi\end{array}}{\text{(Cut)}\ \dfrac{\mathbb{T}[\mathbb{U}[\chi]]\Rightarrow\psi \qquad \begin{array}{c}\vdots\\ \mathbb{V}\Rightarrow\chi\end{array}}{\mathbb{T}[\mathbb{U}[\mathbb{V}]]\Rightarrow\psi}}$$

Nevertheless, such a proof can be transformed into

$$\text{(Cut)}\ \frac{\begin{array}{c}\vdots\\ \mathbb{T}[(\mathbb{U}[\chi],\mathbb{U}[\chi])]\Rightarrow\psi\end{array} \qquad \begin{array}{c}\vdots\\ \mathbb{V}\Rightarrow\chi\end{array}}{\text{(Cut)}\ \dfrac{\mathbb{T}[(\mathbb{U}[\mathbb{V}],\mathbb{U}[\chi])]\Rightarrow\psi \qquad \begin{array}{c}\vdots\\ \mathbb{V}\Rightarrow\chi\end{array}}{\text{(c)}\ \dfrac{\mathbb{T}[(\mathbb{U}[\mathbb{V}],\mathbb{U}[\mathbb{V}])]\Rightarrow\psi}{\mathbb{T}[\mathbb{U}[\mathbb{V}]]\Rightarrow\psi}}}$$

Thus we can prove a variant of Theorem A.2.2 assuming a more general definition of (Cut) that makes possible to perform these two cuts at once, see [HOS94], and a small change in the inductive argument.[1]

Therefore it is sufficient to show that Corollary A.5.4 holds even with exchange and contraction.

[1] It is worth noting that we have the rule of contraction for structures and not only for formulae. The rule of contraction for formulae does not admit cut-elimination. However, as we have product in the language both rules are equivalent in the presence of (Cut), cf. [GO10].

Exchange

As all our simple formulae can be represented as sequences of propositional variables, where brackets tight to right, it is a simple matter to show that our construction works when the rule of exchange is added, because in our case we can define a normal form. For example it is clear that $(e((Xe)e'))$ is equivalent to $ee'eX$. We define the function l by

$$\mathrm{l}(\varphi\cdot\psi)=\begin{cases}\varphi\cdot\mathrm{l}(\psi) & \text{if } \psi \text{ is not a propositional variable,}\\ \psi\cdot\mathrm{l}(\varphi) & \text{if } \varphi \text{ is not a propositional variable,}\\ \varphi\cdot\psi & \text{if } \psi \text{ is a propositional variable which is a capital letter,}\\ \psi\cdot\varphi & \text{if } \varphi \text{ is a propositional variable which is a capital letter,}\end{cases}$$

and $\mathrm{l}(p)=p$ for any propositional variable p. We can change the definition of simple representation, see Definition A.2.4, not to be a join of simple formulae χ_i, but a join of simple formulae $\mathrm{l}(\chi_i)$. Then it is easy to check that Lemmata A.5.1–A.5.3 hold with this adapted definition.

Contraction

The case when the rule of contraction is added is easy, because this rule is not applicable in the proofs of Lemmata A.5.1–A.5.3. The reason is that no formula occurring in our construction has $\varphi\cdot\varphi$ as a subformula, cf. the definition of $\Phi[\mathcal{A},\pi]$, $\Gamma_{\mathcal{A}}$, and $\Lambda_{\mathcal{A}}$. Moreover, the rule of exchange has no effect on that.

A.6.2 One-variable fragment

It is easily seen that in our construction we can encode any finite sequence of variables by one variable using non-associativity. Let p be the only variable and we want to encode the sequence of variables $p_0,\dots,p_m$. The variable p_i is uniquely determined by i and hence also by $2i+1$.[2] We can represent p_i by the binary representation of $2i+1$, hence we use $k=\lceil\log_2(m+1)\rceil+1$ bits, where we replace all zeros by p and ones by (pp) and all the parentheses in the resulting formula tight to right.

Example A.6.1. If we have $p_0,\dots,p_{15}$ then we need 5 bits. Therefore p_2 is represented by 00101 and hence by $(p(p((pp)(p(pp)))))$.

[2]The reason why we use $2i+1$ instead of i is that this encoding works even with the rule of exchange, because 1 is always the last symbol in the binary representation.

It is obvious that this encoding does not work with the rule of contraction. However, it is easy to obtain a function similar to l from Section A.6.1 which works nicely with the rule of exchange.

A.7 Remarks on algebraic consequences

As **FNL** is complete with respect to lattice-ordered residuated groupoids, see e.g. [GO10], our paper proves that the word problem for them is undecidable. However, we can obtain a stronger result as we need only $\{\cdot, \vee\}$. Let $\mathcal{G}$ be a set with a groupoid operation (product) and join on it, where join is idempotent, commutative, associative, and product distributes over join, i.e. they satisfy equalities in Section A.2.2. As $\langle \mathcal{G}, \vee \rangle$ is a join-semilattice we can define $x \leq y$ iff $y = x \vee y$. Our construction shows that the word problem for such a structure is generally undecidable. Therefore it does not have the finite embedability property (FEP).[3]

The translation is fairly straightforward. We know that $\mathbb{S} \Rightarrow \varphi$ is provable iff $\rho(\mathbb{S}) \Rightarrow \varphi$ is provable. We can translate that into $\rho(\mathbb{S}) \leq \varphi$, i.e. $\varphi = \rho(\mathbb{S}) \vee \varphi$. This way we can translate all the axioms from $\Phi[\mathcal{A}, \pi]$, see Definition A.3.2, and use them as a theory. The 2-tag system given by $\mathcal{A}$ and π terminates on a word $w \in \mathcal{A}^*$ iff $eX \vee eA = e\delta(w)X \vee eX \vee eA$ is provable from this theory.

A direct proof that this works can be based on Sections A.4 and A.5. The correctness is proved as in Section A.4. The proof of completeness by induction on the length of the derivation starting from $eX \vee eA$ is analogous to the proof in Section A.5. Note that our simple representation is convenient for this purpose.

Let us also note that the structural rules of exchange and contraction given in Section A.6.1 correspond to $x \cdot y = y \cdot x$ and $x \leq x \cdot x$, respectively. Therefore the word problem is undecidable even if these equalities hold.

On the other hand, it is easy to check that in the proof of correctness we do not need the idempotency and commutativity of join in the full generality. It suffices to add the following particular equality

$$eA \vee eX = eX \vee eA. \tag{A.31}$$

[3] On the contrary, the FEP holds for distributive lattice-ordered residuated groupoids, see [BF09; Far08; HH14]. There is a quasi-identity in the language $\{\cdot, \vee\}$ that holds in the distributive case but does not hold in the non-distributive one. A join-semilattice L is *distributive* if $a \leq a_1 \vee a_2$, for $a, a_1, a_2 \in L$, implies $a = a'_1 \vee a'_2$, for some $a'_1, a'_2 \in L$, where $a'_1 \leq a_1$ and $a'_2 \leq a_2$, see [Grä11]. A particular example of such a quasi-identity is $x \leq x_1 \vee x_2$ and $y \leq y_1 \vee y_2$ implies $xy \leq xy_1 \vee x_1y_2 \vee x_2y$, which was kindly provided by Rostislav Horčík.

However, for our construction to work we need the associativity of join and the distributivity of product over join.

A.8 Remarks on term rewriting systems

A natural way how to think about our construction is that we produce a term rewriting system. For our purposes here, a term rewriting system is a collection of *rewriting rules* $\varphi \to \psi$, where φ and ψ are formulae. Such a rule says that in a formula we can replace any of its subformula φ by ψ.

It is clear that we can read $\varphi \Rightarrow \psi$ as $\varphi \to \psi$. Therefore our rewriting system is given by $\Phi[\mathcal{A}, \pi]$ and the rules of the sequent calculus. Note that strictly speaking rewriting rules are stronger than (Cut), but in our case it is a simple matter to simulate them by (Cut).

An obvious question to ask is whether all rules given by the sequent calculus are necessary for the proof of correctness in Section A.4. It is easily seen that we can replace the sequent calculus with the following rules. For any formulae φ, ψ, and χ we need

$$\varphi \cdot (\psi \vee \chi) \to (\varphi \cdot \psi) \vee (\varphi \cdot \chi), \tag{A.32}$$

$$(\varphi \vee \psi) \cdot \chi \to (\varphi \cdot \chi) \vee (\psi \cdot \chi). \tag{A.33}$$

Moreover, we do not need the full idempotency, commutativity, and associativity of join. It is sufficient to add the particular instance

$$(eX \vee eA) \vee eA \to eX \vee eA \tag{A.34}$$

to $\Phi[\mathcal{A}, \pi]$. It follows that the *accessibility problem* $\varphi \to^* \psi$, whether we can rewrite φ to ψ in finitely many steps, is generally undecidable even for rewriting systems satisfying only rules (A.32) and (A.33). A particular undecidable problem is $e\delta(w)X \vee eA \to^* eX \vee eA$.

It follows easily from the previous sections that the accessibility problem remains generally undecidable if we add any (general) rules which are subsumed by the commutativity and contractivity of product; the idempotency, commutativity, and associativity of join; the distributivity of product over join (in the opposite direction).

Acknowledgements

The author would like to thank to Rostislav Horčík who identified the $\{\cdot, \vee\}$-fragment as the important one. He was also very helpful and supportive during the author's work on the problem. The anonymous referee, Zuzana

Haniková, and Marta Bílková had many useful comments on various versions of this paper. Félix Bou asked the question whether the construction works in the one-variable fragment. The work was supported by grant P202/10/1826 of the Czech Science Foundation and by the long-term strategic development financing of the Institute of Computer Science (RVO:67985807).

Bibliography

[BA05] Willem J. Blok and Clint J. van Alten. 'On the finite embeddability property for residuated ordered groupoids'. *Transactions of the AMS* 357.10 (2005), pp. 4141–4157. DOI: 10.1090/S0002-9947-04-03654-2.

[BF09] Wojciech Buszkowski and Maciej Farulewski. 'Nonassociative Lambek Calculus with Additives and Context-Free Languages'. In: *Languages: From Formal to Natural*. Ed. by Orna Grumberg, Michael Kaminski, Shmuel Katz and Shuly Wintner. Lecture Notes in Computer Science 5533. Berlin Heidelberg: Springer, 2009, pp. 45–58. DOI: 10.1007/978-3-642-01748-3_4.

[Bus05] Wojciech Buszkowski. 'Lambek Calculus with Nonlogical Axioms'. In: *Language and Grammar: Studies in Mathematical Linguistics and Natural Language*. Ed. by C. Casadio, P. J. Scott and R. A. G. Seely. CSLI Lecture Notes 168. Stanford: CSLI, 2005, pp. 77–93.

[Bus82] Wojciech Buszkowski. 'Some Decision Problems in the Theory of Syntactic Categories'. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 28.33-38 (1982), pp. 539–548. ISSN: 1521-3870. DOI: 10.1002/malq.19820283308.

[CM64] John Cocke and Marvin Lee Minsky. 'Universality of Tag Systems with P=2'. *Journal of the ACM* 11.1 (1964), pp. 15–20. DOI: 10.1145/321203.321206.

[Doš88] Kosta Došen. 'Sequent-systems and groupoid models. I'. English. *Studia Logica* 47.4 (1988), pp. 353–385. ISSN: 0039-3215. DOI: 10.1007/BF00671566.

[Far08] Maciej Farulewski. 'Finite embeddability property for residuated groupoids'. *Reports on Mathematical Logic* 43 (2008), pp. 25–42. URL: http://rml.tcs.uj.edu.pl/rml-43/02-farulewski.pdf.

[Gal+07] Nikolaos Galatos, Peter Jipsen, Tomasz Kowalski and Hiroakira Ono. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*. Vol. 151. Studies in Logic and the Foundations of Mathematics. Amsterdam: Elsevier, Apr. 2007, p. 532.

[GO10] Nikolaos Galatos and Hiroakira Ono. 'Cut elimination and strong separation for substructural logics: An algebraic approach'. *Annals of Pure and Applied Logic* 161.9 (2010), pp. 1097–1133. DOI: 10.1016/j.apal.2010.01.003.

[Grä11] George Grätzer. *Lattice Theory: Foundation*. Basel: Birkhäuser, 2011.

[HH14] Zuzana Haniková and Rostislav Horčík. 'Finite Embeddability Property for Residuated Groupoids'. *Algebra Universalis* 72.1 (2014), pp. 1–13. DOI: 10.1007/s00012-014-0284-1.

[HOS94] Ryuichi Hori, Hiroakira Ono and Harold Schellinx. 'Extending intuitionistic linear logic with knotted structural rules.' *Notre Dame Journal of Formal Logic* 35.2 (Mar. 1994), pp. 219–242. DOI: 10.1305/ndjfl/1094061862.

[Lam58] Joachim Lambek. 'The Mathematics of Sentence Structure'. *American Mathematical Monthly* 65.3 (1958), pp. 154–170. URL: http://www.jstor.org/stable/2310058.

[Lam61] Joachim Lambek. 'On the calculus of syntactic types'. In: *Structure of Language and Its Mathematical Aspects*. Ed. by Roman Jakobson. Providence, Rhode Island: American Mathematical Society, 1961, pp. 166–178. DOI: 10.1090/psapm/012.

[Ono98] Hiroakira Ono. 'Proof-Theoretic Methods in Nonclassical Logic: An Introduction'. In: *Theories of Types and Proofs*. Ed. by M. Takahashi, M. Okada and M. Dezani-Ciancaglini. Vol. 2. MSJ Memoirs. Tokyo: Mathematical Society of Japan, 1998, pp. 207–254.

[Pos43] Emil Leon Post. 'Formal Reductions of the General Combinatorial Decision Problem'. *American Journal of Mathematics* 65.2 (1943), pp. 197–215. URL: http://www.jstor.org/stable/2371809.

[Tro92] Anne Sjerp Troelstra. *Lectures on Linear Logic*. CSLI lecture notes 29. Stanford, CA: Stanford, CA : Center for the Study of Language and Information, 1992.

Appendix B

Full Lambek Calculus with Contraction is Undecidable[†]

Karel Chvalovský and Rostislav Horčík

Institute of Computer Science, Academy of Sciences of the Czech Republic,
Pod Vodárenskou věží 271/2, 182 07 Prague 8, Czech Republic

Abstract

We prove that the set of formulae provable in the full Lambek calculus with the structural rule of contraction is undecidable. In fact, we show that the positive fragment of this logic is undecidable.

B.1 Introduction

Besides the cut rule, Gentzen's sequent calculus **LJ** for propositional intuitionistic logic contains other structural rules, namely the rule of contraction (c), exchange (e), left weakening (i), and right weakening (o). By removing all these rules from **LJ**, one arrives at the full Lambek calculus **FL**. More generally, every extension of **FL** by a subset of the rules (c), (e), (i), and (o) defines a logic between **FL** and **LJ**. In [Gal+07] these logics are called *basic substructural logics*. It is known that each of these logics has an analytic sequent calculus. In particular, the cut rule is eliminable in all these calculi if the contraction rule is introduced in its global variant (see [SO96] or [Gal+07, Chapter 4]).

[†]Karel Chvalovský and Rostislav Horčík. 'Full Lambek Calculus with Contraction is Undecidable'. *Journal of Symbolic Logic* (to appear).

Cut elimination is closely related to decidability. It is known that all basic substructural logics are decidable except of $\mathbf{FL_c}$ and $\mathbf{FL_{co}}$, where the former is the extension of $\mathbf{FL}$ by the contraction rule and the latter is the extension of $\mathbf{FL_c}$ by the right weakening rule. The decidability of basic substructural logics without the contraction rule follows immediately from the cut elimination theorem and is proved in [Kom86]. On the other hand, such an easy argument is not applicable for logics with the contraction rule, since this rule makes the proof-search tree infinite. Nevertheless, intuitionistic logic is decidable [Gen35a; Gen35b] and the same holds for the extension of $\mathbf{FL}$ by the exchange and contraction rule [KO91] (the original combinatorial idea from the proof goes back to Kripke [Kri59]). In contrast, we show that $\mathbf{FL_c}$ and $\mathbf{FL_{co}}$ are the only undecidable logics among all basic substructural logics. Actually, we prove that their common positive fragment $\mathbf{FL_c^+}$ is undecidable.

Among known propositional substructural logics, there are not so many logics with an undecidable set of provable formulae. One of them is the relevance logic $\mathbf{R}$, which is a fragment of the involutive distributive $\mathbf{FL}$ with the exchange and contraction rule. The undecidability of its positive fragment is established in [Urq84]. Another example is the extension of $\mathbf{FL}$ by the modular law. Since the equational theory of modular lattices is undecidable [Fre80], one can easily extend this result to the extension of $\mathbf{FL}$ by the modular law, as pointed out in [JT02]. One should also mention the undecidability of propositional linear logic [Lin+92]. Nevertheless, its undecidability is caused by the expressive power of exponentials, while the fragment of linear logic without exponentials is PSPACE-complete [Lin+92].

The following paragraphs outline our undecidability proof. We start with an undecidable problem P from [Hor15] (see Theorem B.3.1), where it is shown that the deducibility problem for $\mathbf{FL_c^+}$ (i.e. the question whether a formula is provable from a finite theory) is undecidable. The problem P is formulated as a reachability question for a string rewriting system simulating a Minsky machine using only square-free words. This ensures that the contraction rule does not affect the simulation of computation.

In order to isolate key ideas of the proof, we refrain from presenting a direct reduction from the problem P into $\mathbf{FL_c^+}$. We instead introduce an auxiliary problem and split the reduction into two steps.

First, Section B.3.2 shows how to reduce the reachability problem for a string rewriting system to the same problem for an atomic conditional string rewriting system, i.e. a string rewriting system where the rules can have only atoms on the right-hand side and their usage is restricted to specific contexts.

Second, an encoding of the reachability problem for an atomic conditional

string rewriting system into $\mathbf{FL}_\mathbf{c}^+$ is presented in Section B.4. A set of rewriting rules is encoded as a lattice conjunction (meet) of implications. The conditionality of rewriting rules is handled by the lattice disjunction (join). The idea of using the lattice disjunction for similar purposes comes from [Kan95], where it is used for linear logic. In fact, our application of this idea was inspired by [Chvar].

The completeness of encoding from Section B.4 is proved by a semantical method similar to the one used in [Laf96]. This method relies on a sound and complete algebraic semantics for $\mathbf{FL}_\mathbf{c}^+$ based on a variety of residuated lattices $\mathcal{RL}_c$. In order not to mix different formalisms, we opt for using an algebraic formalism throughout the paper. Actually, we prove undecidability of the equational theory for $\mathcal{RL}_c$ which immediately implies that $\mathbf{FL}_\mathbf{c}^+$ is undecidable.

Section B.5 contains several comments on possible modifications of the main result, as well as its connection to the deduction theorem.

B.2 Preliminaries

As was mentioned in the introduction, we show that even the positive fragment of full Lambek calculus with the contraction rule $\mathbf{FL}_\mathbf{c}^+$ is undecidable. Probably the most natural way of presenting $\mathbf{FL}_\mathbf{c}^+$ is in terms of a sequent calculus. Formulae are formed in a standard way from a countable set of variables *Var* and a constant 1 using the following connectives: fusion ($\cdot$), two implications ($\backslash$ and $/$), join ($\vee$), and meet ($\wedge$). It should be noted that we have two implications, because there are two natural ways how to obtain them in systems where the rule of exchange is not valid. The set of all formulae is denoted by *Fm*. When writing formulae, we omit some parentheses using the convention that fusion binds tighter than implications followed by meet and join. Furthermore, we use the fact that fusion is associative in $\mathbf{FL}_\mathbf{c}^+$. Moreover, we often omit fusion completely, i.e. a formula $\varphi \cdot \psi$ is shortly written as $\varphi\psi$.

A sequent is a pair $\Gamma \Rightarrow \varphi$, where Γ is a (possibly empty) sequence of formulae and φ is a formula. The elements of Γ are separated by commas as usual and the intended meaning of these commas is fusion.

Definition B.2.1. The sequent calculus for $\mathbf{FL}_\mathbf{c}^+$ has the following axioms and inference rules[1]:

[1]It is worth noting that we formulate the rule of contraction (c) for sequences and not only for formulae. The rule of contraction only for formulae does not admit cut elimination. However, as we have fusion in the language, both rules are equivalent in the presence of (Cut).

$$\text{(Id)}\ \frac{}{\varphi \Rightarrow \varphi} \qquad \text{(Cut)}\ \frac{\Gamma_1, \varphi, \Gamma_2 \Rightarrow \psi \qquad \Delta \Rightarrow \varphi}{\Gamma_1, \Delta, \Gamma_2 \Rightarrow \psi}$$

$$\text{(1L)}\ \frac{\Gamma_1, \Gamma_2 \Rightarrow \psi}{\Gamma_1, 1, \Gamma_2 \Rightarrow \psi} \qquad \text{(1R)}\ \frac{}{\Rightarrow 1}$$

$$\text{($\cdot$L)}\ \frac{\Gamma_1, \varphi, \psi, \Gamma_2 \Rightarrow \chi}{\Gamma_1, \varphi \cdot \psi, \Gamma_2 \Rightarrow \chi} \qquad \text{($\cdot$R)}\ \frac{\Gamma \Rightarrow \varphi \qquad \Delta \Rightarrow \psi}{\Gamma, \Delta \Rightarrow \varphi \cdot \psi}$$

$$\text{($\setminus$L)}\ \frac{\Gamma_1, \varphi, \Gamma_2 \Rightarrow \psi \qquad \Delta \Rightarrow \chi}{\Gamma_1, \Delta, \chi \setminus \varphi, \Gamma_2 \Rightarrow \psi} \qquad \text{($\setminus$R)}\ \frac{\varphi, \Gamma \Rightarrow \psi}{\Gamma \Rightarrow \varphi \setminus \psi}$$

$$\text{($/$L)}\ \frac{\Gamma_1, \varphi, \Gamma_2 \Rightarrow \psi \qquad \Delta \Rightarrow \chi}{\Gamma_1, \varphi / \chi, \Delta, \Gamma_2 \Rightarrow \psi} \qquad \text{($/$R)}\ \frac{\Gamma, \varphi \Rightarrow \psi}{\Gamma \Rightarrow \psi / \varphi}$$

$$\text{($\vee$L)}\ \frac{\Gamma, \varphi, \Delta \Rightarrow \chi \qquad \Gamma, \psi, \Delta \Rightarrow \chi}{\Gamma, \varphi \vee \psi, \Delta \Rightarrow \chi} \qquad \text{($\vee$R)}\ \frac{\Gamma \Rightarrow \varphi_i}{\Gamma \Rightarrow \varphi_1 \vee \varphi_2}\ \text{for } i = 1, 2$$

$$\text{($\wedge$L)}\ \frac{\Gamma, \varphi_i, \Delta \Rightarrow \psi}{\Gamma, \varphi_1 \wedge \varphi_2, \Delta \Rightarrow \psi}\ \text{for } i = 1, 2 \qquad \text{($\wedge$R)}\ \frac{\Gamma \Rightarrow \varphi \qquad \Gamma \Rightarrow \psi}{\Gamma \Rightarrow \varphi \wedge \psi}$$

$$\text{(c)}\ \frac{\Gamma_1, \Delta, \Delta, \Gamma_2 \Rightarrow \psi}{\Gamma_1, \Delta, \Gamma_2 \Rightarrow \psi}$$

The provability in the sequent calculus for $\mathbf{FL}_\mathbf{c}^+$ is defined in the usual way—a proof is a tree labeled by sequents with only axioms in leaves and all the other vertices are obtained from their children by the inference rules. We say that a formula φ is a theorem of $\mathbf{FL}_\mathbf{c}^+$ if $\Rightarrow \varphi$ is provable in $\mathbf{FL}_\mathbf{c}^+$.

The logic $\mathbf{FL}_\mathbf{c}^+$ has a sound and complete algebraic semantics based on residuated lattices. A *residuated lattice* $\mathbf{A} = \langle A, \wedge, \vee, \cdot, \setminus, /, 1\rangle$ is an algebraic structure such that $\langle A, \wedge, \vee\rangle$ is a lattice, $\langle A, \cdot, 1\rangle$ is a monoid and for all $a, b, c \in A$ we have

$$a \cdot b \leq c \quad \text{iff} \quad b \leq a \backslash c \quad \text{iff} \quad a \leq c / b, \tag{B.1}$$

where $\leq$ is the order induced by the lattice structure of $\mathbf{A}$, i.e. $a \leq b$ iff $a \vee b = b$.

Given a residuated lattice $\mathbf{A}$, an $\mathbf{A}$-*evaluation* (or shortly evaluation if $\mathbf{A}$ is clear from the context) e is a map from Fm into A preserving all the connectives, i.e. it is a homomorphism from the formula algebra on Fm to $\mathbf{A}$. An identity is a pair of formulae $\langle \varphi, \psi\rangle$ written suggestively as $\varphi = \psi$. We say that an identity $\varphi = \psi$ holds in $\mathbf{A}$ if for every $\mathbf{A}$-evaluation e we have $e(\varphi) = e(\psi)$. More generally, an identity $\varphi = \psi$ holds in a class of residuated lattices $\mathcal{K}$ if it holds in every residuated lattice from $\mathcal{K}$. An identity of the form $\varphi \vee \psi = \psi$ is shortly denoted by $\varphi \leq \psi$.

Fact B.2.1. *The following identities hold in the class of all residuated lattices:*

- $x(x\backslash y) \leq y$,
- $x(y \vee z) = xy \vee xz$,
- $(y \vee z)x = yx \vee zx$.

Note that $x \leq y$ implies $xz \leq yz$ and $zx \leq zy$ by the distributivity of fusion over join.

A residuated lattice $\mathbf{A}$ is called *square increasing* if the identity $x \leq x^2$ holds in $\mathbf{A}$. It is well known (see e.g. [Gal+07]) that the class of square-increasing residuated lattices forms a variety denoted by $\mathcal{RL}_c$, i.e. it is axiomatized by a set of identities.

Theorem B.2.2 (e.g. [Gal+07])**.** $\mathbf{FL}_\mathbf{c}^+$ *is sound and complete with respect to the class of square-increasing residuated lattices. More precisely, the sequent* $\psi \Rightarrow \varphi$ *is provable in* $\mathbf{FL}_\mathbf{c}^+$ *iff the identity* $\psi \leq \varphi$ *holds in* $\mathcal{RL}_c$.

Since the sequents $\Rightarrow \varphi$ and $1 \Rightarrow \varphi$ are equivalent in terms of provability in $\mathbf{FL}_\mathbf{c}^+$, it follows that the sequent $\Rightarrow \varphi$ is provable in $\mathbf{FL}_\mathbf{c}^+$ iff the identity $1 \leq \varphi$ holds in $\mathcal{RL}_c$. Furthermore, observe that by (B.1) an identity $\varphi \leq \psi$ holds in $\mathcal{RL}_c$ iff $1 \leq \varphi\backslash\psi$ holds there. Consequently, if we prove that the set of identities of the form $\varphi \leq \psi$ valid in $\mathcal{RL}_c$ is undecidable, then we obtain the same for the set of provable formulae in $\mathbf{FL}_\mathbf{c}^+$.

We opt for using algebraic semantics in our proofs because algebraic notation in this case seems to be more compact. Nevertheless, this choice is not essential and has no influence on the construction itself. Moreover, a reader preferring e.g. proof-theoretical notions can adapt even all the proofs, because the main ideas in them remain the very same.

B.2.1 Residuated frames

In the following paragraphs, we recall residuated frames which will be useful in the construction of a suitable countermodel in the proof of completeness of our encoding. We start with an important example of a residuated lattice called the powerset monoid.

Example B.2.1 (see e.g. [Gal+07])**.** Let $\mathbf{M} = \langle M, \cdot, 1\rangle$ be a monoid. The *powerset monoid* is the residuated lattice $\mathcal{P}(\mathbf{M}) = \langle \mathcal{P}(M), \cap, \cup, \cdot, \backslash, /, \{1\}\rangle$

defined on the powerset of M, where for $X, Y, Z \subseteq M$ the operations are defined as follows:

$$\begin{aligned} X \cdot Y &= \{\, x \cdot y \in M \mid x \in X,\ y \in X \,\}, \\ X \backslash Z &= \{\, y \in M \mid X \cdot \{y\} \subseteq Z \,\}, \\ Z / Y &= \{\, x \in M \mid \{x\} \cdot Y \subseteq Z \,\}. \end{aligned}$$

Note that $1 \in X \backslash Z$ iff $X \subseteq Z$.

Other examples of residuated lattices can be obtained from the powerset monoid $\mathcal{P}(\mathbf{M})$ by considering a suitable closure operator on the poset $\langle \mathcal{P}(M), \subseteq \rangle$. Recall that a closure operator on $\langle \mathcal{P}(M), \subseteq \rangle$ is a map $\gamma \colon \mathcal{P}(M) \to \mathcal{P}(M)$ such that for all $X, Y \subseteq M$ we have

- $X \subseteq \gamma(X)$,
- $\gamma(\gamma(X)) = \gamma(X)$, and
- $X \subseteq Y$ implies $\gamma(X) \subseteq \gamma(Y)$.

A subset $X \subseteq M$ is said to be γ-*closed* if $X = \gamma(X)$. The set of all γ-closed subsets of M is denoted $\mathcal{P}(M)_\gamma$. Recall that $\langle \mathcal{P}(M)_\gamma, \cap, \cup_\gamma \rangle$ forms a complete lattice where $X \cup_\gamma Y = \gamma(X \cup Y)$.

A subset $\mathcal{B} \subseteq \mathcal{P}(M)_\gamma$ of γ-closed sets is said to be a *basis* for γ if every $X \in \mathcal{P}(M)_\gamma$ can be expressed as the intersection of all the basis elements above X, i.e. $X = \bigcap \{\, B \in \mathcal{B} \mid X \subseteq B \,\}$. Given a basis $\mathcal{B}$ for the closure operator γ, the equivalence

$$X \subseteq \gamma(Y) \quad \text{iff} \quad Y \subseteq B \text{ implies } X \subseteq B \text{ for all } B \in \mathcal{B} \tag{B.2}$$

holds for all $X, Y \subseteq M$.

It is well known that every closure operator on $\langle \mathcal{P}(M), \subseteq \rangle$ is induced by a binary relation $N \subseteq M \times T$ for some set T (see e.g. [DP02; Gal+07]). Given such a relation $N \subseteq M \times T$, one can introduce the following two maps which define a Galois connection between $\langle \mathcal{P}(M), \subseteq \rangle$ and $\langle \mathcal{P}(T), \subseteq \rangle$:

$$\begin{aligned} X^{\triangleright} &= \{\, b \in T \mid (\forall x \in X)(x\ N\ b) \,\}, \\ Y^{\triangleleft} &= \{\, a \in M \mid (\forall y \in Y)(a\ N\ y) \,\}. \end{aligned}$$

Lemma B.2.3 (see e.g. [DP02; Gal+07])**.** *The maps* $^{\triangleleft}$ *and* $^{\triangleright}$ *have the following properties:*

- $X \subseteq Y$ *implies* $Y^{\triangleright} \subseteq X^{\triangleright}$ *for* $X, Y \subseteq M$.

- $X \subseteq Y$ *implies* $Y^{\triangleleft} \subseteq X^{\triangleleft}$ *for* $X, Y \subseteq T$.
- $\emptyset^{\triangleleft} = M$ *and* $\emptyset^{\triangleright} = T$.
- $X^{\triangleright\triangleleft\triangleright} = X^{\triangleright}$ *and* $Y^{\triangleleft\triangleright\triangleleft} = Y^{\triangleleft}$ *for* $X \subseteq M$ *and* $Y \subseteq T$.
- *The map* $\gamma_N \colon \mathcal{P}(M) \to \mathcal{P}(M)$ *defined by* $\gamma_N(X) = X^{\triangleright\triangleleft}$ *is a closure operator on* $\langle \mathcal{P}(M), \subseteq \rangle$.
- *The collection* $\{ \{b\}^{\triangleleft} \mid b \in T \}$ *forms a basis for* γ_N.

Let $x_1, \ldots, x_n \in M$. To shorten the notation, we will write $\gamma_N\{x_1, \ldots, x_n\}$ instead of $\gamma_N(\{x_1, \ldots, x_n\})$.

Assume that we have a closure operator γ on the powerset monoid $\mathcal{P}(\mathbf{M})$ described in Example B.2.1. If γ satisfies $\gamma(\gamma(X) \cdot \gamma(Y)) = \gamma(X \cdot Y)$ for all $X, Y \subseteq M$ then γ is called a *nucleus*. In this case one can define a residuated lattice on γ-closed sets. The resulting algebra $\mathcal{P}(\mathbf{M})_\gamma = \langle \mathcal{P}(M)_\gamma, \cap, \cup_\gamma, \cdot_\gamma, \backslash, /, \gamma\{1\} \rangle$, where $X \cup_\gamma Y = \gamma(X \cup Y)$ and $X \cdot_\gamma Y = \gamma(X \cdot Y)$, is a residuated lattice (see e.g. [Gal+07]).

We have mentioned above that every binary relation $N \subseteq M \times T$ induces a closure operator γ_N on $\langle \mathcal{P}(M), \subseteq \rangle$. The following definition gives a sufficient condition on N for γ_N to be in addition a nucleus.

Definition B.2.2 ([GJ13])**.** A residuated frame is a two-sorted structure $\mathbf{W} = \langle \mathbf{M}, T, N \rangle$ where $\mathbf{M} = \langle M, \cdot, 1 \rangle$ is a monoid, T is a set, and $N \subseteq M \times T$ is a nuclear relation, i.e. there exist operations $\backslash\!\backslash \colon M \times T \to T$ and $/\!/ \colon T \times M \to T$ such that

$$x \cdot y \; N \; z \qquad \text{iff} \qquad y \; N \; x \backslash\!\backslash z \qquad \text{iff} \qquad x \; N \; z /\!/ y.$$

Given a residuated frame $\mathbf{W} = \langle \mathbf{M}, T, N \rangle$, the induced closure operator γ_N is a nucleus on the powerset monoid $\mathcal{P}(\mathbf{M})$ (see [GJ13]) indeed. Thus one can define a dual algebra $\mathbf{W}^+$ of the residuated frame $\mathbf{W}$ by letting $\mathbf{W}^+$ to be the residuated lattice $\mathcal{P}(\mathbf{M})_{\gamma_N}$.

Now we present an example of a residuated frame associated with a language L over an alphabet Σ and its dual algebra. This example will be of use later. Given an alphabet Σ, the set of all words (resp. non-empty words) over Σ is denoted Σ^* (resp. Σ^+). Recall that Σ^* together with the concatenation of words and the empty word ε forms the free Σ-generated monoid.

Example B.2.2. Let Σ be an alphabet and $L \subseteq \Sigma^*$ a language. Consider a structure $\mathbf{W}_L = \langle \Sigma^*, \Sigma^* \times \Sigma^*, N \rangle$ where the binary relation $N \subseteq \Sigma^* \times$

$(\Sigma^* \times \Sigma^*)$ is defined by

$$x \; N \; \langle u, v \rangle \quad \text{iff} \quad uxv \in L.$$

It follows that N is nuclear, since for all $x, y, u, v \in \Sigma^*$ we have

$$xy \; N \; \langle u, v \rangle \quad \text{iff} \quad y \; N \; \langle ux, v \rangle \quad \text{iff} \quad x \; N \; \langle u, yv \rangle \quad \text{iff} \quad uxyv \in L.$$

Consequently, $\mathbf{W}_L$ forms a residuated frame and the dual algebra

$$\mathbf{W}_L^+ = \langle W_L^+, \cap, \cup_{\gamma_N}, \cdot_{\gamma_N}, \backslash, /, \gamma_N\{\varepsilon\} \rangle$$

is a residuated lattice where $W_L^+ = \mathcal{P}(\Sigma^*)_{\gamma_N}$, $X \cup_{\gamma_N} Y = \gamma_N(X \cup Y)$ and $X \cdot_{\gamma_N} Y = \gamma_N(X \cdot Y)$.

In what follows, we assume that $\Sigma \subseteq \mathit{Var}$. Hence $\Sigma^* \subseteq \mathit{Fm}$ if we identify a word $a_1 a_2 \dots a_n \in \Sigma^*$ with the formula $a_1 \cdot a_2 \cdot \dots \cdot a_n$ (the fusion of atoms $a_1, \dots, a_n$).

Lemma B.2.4. *Let $e\colon \mathit{Fm} \to W^+$ be a $\mathbf{W}_L^+$-evaluation, $a_1, \dots, a_n \in \Sigma$ and $w = a_1 a_2 \dots a_n$. Assume that $e(a_i) = \gamma_N(X_i)$ and $X_i \subseteq \Sigma^*$ for $i = 1, \dots, n$. Then $e(w) = \gamma_N(X_1 \cdot X_2 \cdot \dots \cdot X_n)$. In particular, if $X_i = \{a_i\}$ for $i = 1, \dots, n$ then $e(w) = \gamma_N\{w\}$.*

Proof. By the definition of a nucleus we have

$$\gamma_N(X) \cdot_{\gamma_N} \gamma_N(Y) = \gamma_N(\gamma_N(X) \cdot \gamma_N(Y)) = \gamma_N(X \cdot Y).$$

This can be easily generalized for arbitrarily many subsets $X_1, \dots, X_n \subseteq \Sigma^*$ and hence

$$\gamma_N(X_1) \cdot_{\gamma_N} \dots \cdot_{\gamma_N} \gamma_N(X_n) = \gamma_N(X_1 \cdots X_n).$$

Therefore the lemma follows, since

$$e(w) = e(a_1) \cdot_{\gamma_N} \dots \cdot_{\gamma_N} e(a_n) = \gamma_N(X_1) \cdot_{\gamma_N} \dots \cdot_{\gamma_N} \gamma_N(X_n) = \gamma_N(X_1 \cdots X_n).$$

□

Certainly, we are interested in languages L such that $\mathbf{W}_L^+$ is square increasing. We say that a language L over an alphabet Σ is closed under the contraction rule if $uxxv \in L$ implies $uxv \in L$ for all $u, x, v \in \Sigma^*$.

Lemma B.2.5. *If $L \subseteq \Sigma^*$ is closed under the contraction rule then $\mathbf{W}_L^+ \in \mathcal{RL}_c$.*

Proof. Let $X \in W_L^+$, i.e. $X = \gamma_N(X)$. It suffices to show that $(X \cdot X)^{\triangleright} \subseteq X^{\triangleright}$, for if this is proved, Lemma B.2.3 gives

$$X = X^{\triangleright\triangleleft} \subseteq (X \cdot X)^{\triangleright\triangleleft} = X \cdot_{\gamma_N} X.$$

Assume that $\langle u, v\rangle \in (X \cdot X)^{\triangleright}$ and $x \in X$. Hence $xx \in X \cdot X$ and so $uxxv \in L$. Since L is closed under the contraction rule, we have $uxv \in L$. Thus $\langle u, v\rangle \in X^{\triangleright}$. □

B.2.2 Regular languages

In what follows, we will need to encode regular languages closed under the contraction rule into the equational theory of $\mathcal{RL}_c$. In this section we will show how to do it. It also affords a good illustration of how to use residuated frames in order to prove a completeness of an encoding.

Given an alphabet Σ, the set of all regular languages over Σ is denoted $\mathrm{Reg}(\Sigma)$. Recall that every regular language can be captured by a right-linear context-free grammar (see [HU79]). Let $G = \langle V, \Sigma, P, S\rangle$ be a right-linear context-free grammar with a finite set of variables V (non-terminals), a finite set of terminals Σ, a start variable S and a finite set of production rules P of the form $A \to wB$ or $A \to w$ for some variables $A, B \in V$ and $w \in \Sigma^*$. The derivation relation $\to_G^*$ of G is defined in the usual way (see e.g. [HU79]). For every non-terminal $A \in V$ we define its language $L(A) = \{\, w \in \Sigma^* \mid A \to_G^* w \,\}$. In particular, $L(S)$ is the language generated by G. Until further notice, we assume that $V \cup \Sigma \subseteq \mathit{Var}$.

We define a finite set of formulae

$$\Delta_G = \{\, wB\backslash A \mid A \to wB \in P \,\} \cup \{\, w\backslash A \mid A \to w \in P \,\}.$$

Then we define a formula δ_G as the meet of 1 and all the formulae in Δ_G, i.e. $\delta_G = 1 \wedge \bigwedge \Delta_G$.

Now we can encode the membership of a word $w \in \Sigma^*$ in the regular language $L = L(S)$ generated by G via the identity $w\delta_G \leq S$. The following lemma shows that this encoding is sound.

Lemma B.2.6. *Let $G = \langle V, \Sigma, P, S\rangle$ be a right-linear context-free grammar generating a regular language L and $w \in (V \cup \Sigma)^*$. If $S \to_G^* w$ then $w\delta_G \leq S$ holds in $\mathcal{RL}_c$. In particular, if $w \in L = L(S)$ then $w\delta_G \leq S$ holds in $\mathcal{RL}_c$.*

Proof. The claim is proved by induction on the number of steps in the derivation of w using the grammar G. Clearly, $S\delta_G \leq S$ holds in $\mathcal{RL}_c$, since $\delta_G \leq 1$. Assume that w is derived by a production rule $A \to uB \in P$,

i.e. $w = w'uB$. Hence $uB\backslash A \in \Delta_G$ and so $\delta_G \leq uB\backslash A$. By the induction hypothesis, we know that $w'A\delta_G \leq S$ holds in $\mathcal{RL}_c$. It follows that

$$w\delta_G \leq w\delta_G^2 \leq w(uB\backslash A)\delta_G = w'uB(uB\backslash A)\delta_G \leq w'A\delta_G \leq S.$$

The case for a production rule of the form $A \to u$ is completely analogous. □

We prove the completeness of our encoding via the residuated frame $\mathbf{W}_L$ (see Example B.2.2). We start with a general lemma.

Lemma B.2.7. *Let $G = \langle V, \Sigma, P, S\rangle$ be a right-linear context-free grammar and $\mathbf{W} = \langle \Sigma^*, T, N\rangle$ a residuated frame. Given a $\mathbf{W}^+$-evaluation $e\colon Fm \to W^+$ suppose that $e(a) = \gamma_N\{a\}$ for $a \in \Sigma$ and $e(A) = \gamma_N(L(A))$ for $A \in V$. Then $\varepsilon \in e(\delta_G)$.*

Proof. We have to show that $\varepsilon \in e(\delta_G) = e(1) \cap \bigcap_{\varphi\in\Delta_G} e(\varphi)$. Since $\varepsilon \in \gamma_N\{\varepsilon\} = e(1)$, it suffices to show that for every $\varphi \in \Delta_G$ we have $\varepsilon \in e(\varphi)$. In other words, we need to show that $e(wB) \subseteq e(A)$ (resp. $e(w) \subseteq e(A)$) if $\varphi = wB\backslash A$ (resp. $\varphi = w\backslash A$).

Assume that $\varphi = wB\backslash A$ (the proof for $\varphi = w\backslash A$ is analogous). Hence the production rule $A \to wB$ belongs to P. Let $x \in L(B)$. Hence $A \to_G wB \to_G^* wx$, i.e. $wx \in L(A)$. Consequently, we have $\{w\} \cdot L(B) \subseteq L(A)$. Thus Lemma B.2.4 implies

$$e(wB) = \gamma_N(\{w\} \cdot L(B)) \subseteq \gamma_N(L(A)) = e(A).$$

□

Assume that the regular language L generated by G is closed under the contraction rule and $w\delta_G \leq S$ holds in $\mathcal{RL}_c$. Hence $\mathbf{W}_L^+$ belongs to $\mathcal{RL}_c$ by Lemma B.2.5. Thus $w\delta_G \leq S$ holds in $\mathbf{W}_L^+$. Consider the evaluation from Lemma B.2.7. It follows that $e(w\delta_G) = \gamma_N(e(w) \cdot e(\delta_G)) \subseteq e(S) = \gamma_N\{L\}$. Since $\varepsilon \in e(\delta_G)$ by Lemma B.2.7 and $w \in \gamma_N\{w\} = e(w)$ by Lemma B.2.4, we obtain $w \in e(w)\cdot e(\delta_G) \subseteq e(w\delta_G) \subseteq \gamma_N\{L\}$. In order to show that $w \in L$, it suffices to show that L is γ_N-closed. To see this observe that $L = \{\langle \varepsilon, \varepsilon\rangle\}^{\triangleleft}$. Thus L is a member of the basis for γ_N (see Lemma B.2.3).

Theorem B.2.8. *Let L be a regular language closed under the contraction rule, $G = \langle V, \Sigma, P, S\rangle$ its generating right-linear context-free grammar and $\delta_G = 1 \wedge \bigwedge \Delta_G$. Then $w \in L$ iff $w\delta_G \leq S$ holds in $\mathcal{RL}_c$.*

B.3 SRSs and atomic conditional SRSs

A *string rewriting system* (shortly SRS) is a tuple $\langle \Sigma, R \rangle$, where Σ is an alphabet and $R \subseteq \Sigma^* \times \Sigma^*$ is a binary relation. A member $\langle x, y \rangle$ of R is called a (rewriting) rule and we write it $x \rightsquigarrow y$.

A single-step reduction relation $\to \subseteq \Sigma^* \times \Sigma^*$ is defined for any $w, w_1 \in \Sigma^*$ as $w \to w_1$ iff there are words $x, y, u, v \in \Sigma^*$ such that $w = uxv$, $w_1 = uyv$ and $x \rightsquigarrow y \in R$. A reduction relation $\to^*$ is the reflexive transitive closure of $\to$. For further details see e.g. [BO93].

Given a word $w_0 \in \Sigma^*$, we define a language corresponding to $\langle \Sigma, R \rangle$ and w_0 by

$$L(w_0) = \{\, w \in \Sigma^* \mid w \to^* w_0 \,\}.$$

The problem to decide whether a word $w \in \Sigma^*$ belongs to $L(w_0)$ is sometimes called the reachability problem (for a fixed word w_0).

It is easy to construct an SRSs $\langle \Sigma, R \rangle$ and $w_0 \in \Sigma^+$ such that $L(w_0)$ is undecidable. A common way how to obtain such a rewriting system is to encode a Minsky machine (two-counter machine) with an undecidable set of accepting configurations. These machines have a finite set of states and therefore their configuration can be completely described by a triplet $\langle i, m, n \rangle$ of natural numbers, which says that the machine is in the state i and counters have values m and n. A possible way how to encode such a triplet by a word is

$$Aa^m q_i a^n B,$$

where A, B are stoppers and a^k is the sequence of k letters a. One can also capture operations of Minsky machines by rewriting rules. It follows that the language $L(Aq_0B)$ is undecidable—a Minsky machine accepts a configuration $\langle i, m, n \rangle$ if its computation ends in $\langle 0, 0, 0 \rangle$. However, in our case the problem is that $L(Aq_0B)$ is not closed under the contraction rule. Therefore such a straightforward representation of counters is impossible.

Nevertheless, we can represent counters by square-free words, i.e. do not contain uu as a subword. It is well known (see e.g. [Lot02]) that if we have a morphism over the alphabet $\{a, b, c\}$ defined by

$$h(a) = abc \qquad h(b) = ac \qquad h(c) = b$$

then $h^m(a)$ is square free for any m. Hence we can represent a state of our machine by

$$Ah^m(a)Bq_iCh^n(a)D.$$

In this way we can obtain a rewriting system such that the language $L(AaBq_0CaD)$ is undecidable and consists only of square-free words and

therefore it is closed under the contraction rule. This coding is inspired by [KS95, Section 7.2.5] and the complete construction is described in [Hor15, Section 4], where the word problem for $\mathcal{RL}_c$ and therefore the deducibility problem for $\mathbf{FL}_\mathbf{c}^+$ are proved to be undecidable.

Theorem B.3.1 ([Hor15])**.** *There is an SRS $\langle \Sigma, R\rangle$ and $w_0 \in \Sigma^+$ such that $L(w_0)$ is undecidable. In addition, $L(w_0)$ consists only of square-free words, i.e. $L(w_0)$ is closed under the contraction rule. Moreover, the rules contain only square-free words.*

It is worth pointing out that R from the previous theorem contains only non-empty words, i.e. $R \subseteq \Sigma^+ \times \Sigma^+$, and hence if $w \to^* w_0$ then $w \in \Sigma^+$, since $w_0 \in \Sigma^+$. Clearly, empty words play no essential role in this SRS,[2] the fact to be used later.

In this paper we will present an encoding of the reduction relation $\to^*$ for such a rewriting system into the equational theory of $\mathcal{RL}_c$.

B.3.1 A naïve way of encoding

Theorem B.3.1 gives us an SRS $\langle \Sigma, R\rangle$ and $w_0 \in \Sigma^+$ such that it is undecidable whether $w \to^* w_0$ for $w \in \Sigma^+$. Algebraically we would like to encode this problem as the validity of $w \leq w_0$ modulo the given set of rules R. The problem is how to represent the set of rules R. Now we are going to present a naïve way how to do that. Although it does not work, we will elaborate on it later on.

The most natural way is to represent a rule $x \rightsquigarrow y \in R$ as an implication, e.g. $x\backslash y$. The idea is as follows. Assume we have $uxv, uyv \in \Sigma^*$. We know $x(x\backslash y) \leq y$ (see Fact B.2.1) and hence $ux(x\backslash y)v \leq uyv$. Moreover, as we have contraction, it holds that

$$ux(x\backslash y)v \leq ux(x\backslash y)(x\backslash y)v \leq uy(x\backslash y)v.$$

This shows how to represent a rewriting rule, but we can easily generalize this to the whole set R using meet. Let $\theta = \bigwedge_{x \rightsquigarrow y \in R}(x\backslash y)$ then in the previous example we have

$$ux\theta v \leq ux\theta\theta v \leq ux(x\backslash y)\theta v \leq uy\theta v.$$

However, such a straightforward representation fails. Assume also $z \rightsquigarrow xv \in R$. We have $uz \to uxv \to uyv$. Hence we would like to show that $uz\theta \leq ux\theta v \leq$

[2]For these reasons even the definition of SRSs in [Hor15] allows only non-empty words.

$uy\theta v$, but the previous technique does not work. It is not enough that $z\backslash xv$ is in θ, because we would need $z\backslash x\theta v$ to be in θ, which is obviously impossible, since it cannot contain itself.

Obviously, this is an essential problem. If all $x \rightsquigarrow y \in R$ were such that y is only a letter, i.e. $y \in \Sigma$, then this would work, but for obvious reasons there is no such SRS satisfying Theorem B.3.1. However, it is possible to define a modification of SRSs (Section B.3.2) such that the naïve way of encoding will be applicable on these modified systems (Section B.4).

B.3.2 Conditional string rewriting systems

To overcome the problem with the naïve encoding, we introduce a certain modification of string rewriting systems, which we call conditional SRSs. A *conditional string rewriting system* (or CSRS) is a tuple $\langle \Sigma, R \rangle$, where Σ is an alphabet and $R \subseteq \Sigma^* \times \Sigma^* \times \mathrm{Reg}(\Sigma) \times \mathrm{Reg}(\Sigma)$ is a relation. A member $\langle x, y, L_\ell, L_r \rangle$ of R consists of a rewriting rule $x \rightsquigarrow y$ and two regular languages L_ℓ, L_r and expresses the fact that the rule $x \rightsquigarrow y$ can be used only in a context restricted by the languages L_ℓ, L_r. We denote the tuple $\langle x, y, L_\ell, L_r \rangle$ more suggestively $\langle x \rightsquigarrow y, L_\ell, L_r \rangle$. A single-step reduction relation $\to \subseteq \Sigma^* \times \Sigma^*$ is defined for any $w, w_1 \in \Sigma^*$ by

$$w \to w_1 \quad \text{iff} \quad \text{there are a rule } \langle x \rightsquigarrow y, L_\ell, L_r \rangle \in R \text{ and words } u \in L_\ell \text{ and } v \in L_r \text{ such that } w = uxv \text{ and } w_1 = uyv.$$

A reduction relation $\to^*$ is the reflexive transitive closure of $\to$.

Note that similar rewriting systems were considered in the literature. For instance, Chottin in [Cho79] defined so-called controlled string rewriting systems where only left contexts are restricted by regular languages.

A CSRS is said to be *atomic* if all its rules have atomic right-hand sides, i.e. if for every $\langle x \rightsquigarrow y, L_\ell, L_r \rangle$ in R we have $y \in \Sigma$.

In the rest of this section we are going to show that every SRS $\langle \Sigma, R \rangle$, which has $R \subseteq \Sigma^* \times \Sigma^+$,[3] can be simulated by an atomic CSRS. More precisely, assume that we have another two copies of Σ denoted $\Sigma' = \{ a' \mid a \in \Sigma \}$ and $\Sigma'' = \{ a'' \mid a \in \Sigma \}$ such that Σ, Σ' and Σ'' are disjoint. We will prove that there is an atomic CSRS $\langle \Sigma \cup \Sigma' \cup \Sigma'', R' \rangle$ such that for every $w, w_0 \in \Sigma^*$ we have $w \to^* w_0$ iff $w \rightsquigarrow^*_{R'} w_0$.

Clearly, every rule $x \rightsquigarrow a$ in R with an atomic right-hand side can be simulated by the atomic conditional rule $\langle x \rightsquigarrow a, \Sigma^*, \Sigma^* \rangle$. Every non-atomic

[3] This assumption is useful for simplifying the construction, since the SRS from Theorem B.3.1 satisfies it. However, we could extend the construction even to rules of the form $x \rightsquigarrow \varepsilon$. Roughly speaking, it would suffice to allow empty words in the definition of atomic CSRSs and handle such rules similarly to atomic rules.

rule $x \rightsquigarrow a_1 \dots a_n$ from R, where $n \geq 2$ and $a_1, \dots, a_n \in \Sigma$, is simulated by the following atomic conditional rules:

$$\langle \varepsilon \rightsquigarrow a_i'', \Sigma^*(\Sigma'')^*, \Sigma^* \rangle \qquad \text{for } i \in \{2, \dots, n\}, \tag{B.3}$$
$$\langle x \rightsquigarrow a_1', \Sigma^*, a_2'' \dots a_n'' \Sigma^* \rangle, \tag{B.4}$$
$$\langle a_i'' \rightsquigarrow a_i, \Sigma^*\Sigma'(\Sigma'')^*, \Sigma^* \rangle \qquad \text{for } i \in \{2, \dots, n\}, \tag{B.5}$$
$$\langle a_1' \rightsquigarrow a_1, \Sigma^*, \Sigma^* \rangle. \tag{B.6}$$

Lemma B.3.2. *Let $w, w_0 \in \Sigma^*$. Then $w \to^* w_0$ implies $w \rightsquigarrow_{R'}^* w_0$.*

Proof. By induction on the length of derivation. The simulation of a rule $x \rightsquigarrow a$ with an atomic right-hand side is obvious. Let $x \rightsquigarrow a_1 \dots a_n$ be a rule in R having a non-atomic right-hand side. The rewriting step $uxv \to ua_1 \dots a_n v$ for $u, v \in \Sigma^*$ is simulated as follows:

$$\begin{aligned}
uxv &\rightsquigarrow_{R'} uxa_2''v && \text{by (B.3)} \\
&\rightsquigarrow_{R'}^* uxa_2'' \dots a_n''v && \text{by (B.3)} \\
&\rightsquigarrow_{R'} ua_1'a_2'' \dots a_n''v && \text{by (B.4)} \\
&\rightsquigarrow_{R'}^* ua_1'a_2 \dots a_n v && \text{by (B.5)} \\
&\rightsquigarrow_{R'} ua_1a_2 \dots a_n v && \text{by (B.6).}
\end{aligned}$$

□

For the converse direction we need to interpret the auxiliary words containing symbols from Σ' and Σ'' back in Σ^*. For this purpose we define two monoid homomorphisms

$$h_1 \colon (\Sigma \cup \Sigma' \cup \Sigma'')^* \to \Sigma^* \quad \text{and} \quad h_2 \colon (\Sigma \cup \Sigma'')^* \to \Sigma^*$$

respectively by

$$h_1(a) = h_1(a') = h_1(a'') = a \quad \text{and} \quad h_2(a) = a,\ h_2(a'') = \varepsilon$$

for all $a \in \Sigma$.

Since the domains of h_1 and h_2 are the free monoids, the above definitions extend uniquely to the whole domains. Then we merge the above homomorphisms together and define a mapping $h \colon (\Sigma \cup \Sigma' \cup \Sigma'')^* \to \Sigma^*$ by

$$h(w) = \begin{cases} h_2(w) & \text{if } w \in (\Sigma \cup \Sigma'')^*, \\ h_1(w) & \text{otherwise, i.e. } w \text{ contains a letter from } \Sigma'. \end{cases}$$

Note that $h(w) = h_2(w) = w$ for $w \in \Sigma^*$.

Lemma B.3.3. *If $w \rightsquigarrow^*_{R'} w_0$ then $h(w) \to^* h(w_0)$ for $w, w_0 \in (\Sigma \cup \Sigma' \cup \Sigma'')^*$. In particular, $w \rightsquigarrow^*_{R'} w_0$ implies $w \to^* w_0$ for $w, w_0 \in \Sigma^*$.*

Proof. By induction on the length of derivation. Assume that $w \rightsquigarrow_{R'} w_1 \rightsquigarrow^*_{R'} w_0$. By the induction hypothesis, we have $h(w_1) \to^* h(w_0)$. If $w = uxv$, $w_1 = uav$ for $u, v, x \in \Sigma^*$, $a \in \Sigma$, and $\langle x \rightsquigarrow a, \Sigma^*, \Sigma^* \rangle \in R'$ then $x \rightsquigarrow a \in R$ and the lemma holds trivially. Assume that the rule from R' used in $w \rightsquigarrow_{R'} w_1$ is among the rules (B.3)–(B.6) corresponding to a rule $x \rightsquigarrow a_1 \dots a_n$ in R. The proof splits into two cases.

First, suppose that $w \in (\Sigma \cup \Sigma'')^*$. Hence $h(w) = h_2(w)$. Note that only the rule (B.3) or (B.4) can be applied to w. If (B.3) is applied then $w = uv$ and $w_1 = ua''_i v$ for some $u \in \Sigma^*(\Sigma'')^*$, $v \in \Sigma^*$, and $i \in \{2, \dots, n\}$. Thus $w_1 \in (\Sigma \cup \Sigma'')^*$ and we have

$$h(w_1) = h_2(w_1) = h_2(ua''_i v) = h_2(uv) = h(w).$$

Therefore $h(w) \to^* h(w_1)$ by reflexivity.

If (B.4) is applied then $w = uxa''_2 \dots a''_n v$ and $w_1 = ua'_1 a''_2 \dots a''_n v$ for some $u, v, x \in \Sigma^*$. Hence

$$h(w) = h_2(w) = h_2(uxa''_2 \dots a''_n v) = uxv$$

and

$$h(w_1) = h_1(ua'_1 a''_2 \dots a''_n v) = ua_1 a_2 \dots a_n v.$$

Consequently, we have $h(w) = uxv \to ua_1 a_2 \dots a_n v = h(w_1)$.

Second, suppose that $w \notin (\Sigma \cup \Sigma'')^*$, i.e. w contains a letter $a' \in \Sigma'$. Hence $h(w) = h_1(w)$ and only the rule (B.5) or (B.6) can be applied to w. If (B.5) is applied then $w = ua'z''a''_i v$ and $w_1 = ua'z''a_i v$ for some $u, v \in \Sigma^*$, $a' \in \Sigma'$, $z'' \in (\Sigma'')^*$, and $i \in \{2, \dots, n\}$. This gives

$$h(w) = h_1(w) = h_1(ua'z''a''_i v) = h_1(ua'z'')a_i h_1(v) = h_1(ua'z''a_i v) = h(w_1)$$

and so $h(w) \to^* h(w_1)$.

Finally, if (B.6) is applied then $w = ua'_1 v$ and $w_1 = ua_1 v$ for some $u, v \in \Sigma^*$. We thus get $h(w) = h_1(w) = ua_1 v = h_2(w_1) = h(w_1)$ and so $h(w) \to^* h(w_1)$. □

Assume that the language $L(w_0)$ corresponding to the original SRS $\langle \Sigma, R \rangle$ consists of square-free words only. The next lemma shows that the language

$$L'(w_0) = \{\, w \in (\Sigma \cup \Sigma' \cup \Sigma'')^* \mid w \rightsquigarrow^*_{R'} w_0 \,\}$$

associated with the atomic CSRS $\langle \Sigma \cup \Sigma' \cup \Sigma'', R' \rangle$ also contains only square-free words.

Lemma B.3.4. *The language $L'(w_0) \subseteq (\Sigma \cup \Sigma' \cup \Sigma'')^*$ contains only square-free words.*

Proof. Let $w \in L'(w_0)$. We prove the lemma by induction on the length of derivation. The base case is easy, since $w_0 \in L(w_0)$ is a square-free word. Suppose that $w \rightsquigarrow_{R'} w_1 \rightsquigarrow^*_{R'} w_0$. By the induction hypothesis, w_1 is square free. We distinguish five cases according to the rule used in $w \rightsquigarrow_{R'} w_1$. Before that note the following general fact. Since $w \in L'(w_0)$, we have $h(w) \to^* h(w_0) = w_0$ by Lemma B.3.3. Hence $h(w) \in L(w_0)$.

If a rule $\langle x \rightsquigarrow a, \Sigma^*, \Sigma^* \rangle \in R'$ corresponding to an atomic rule $x \rightsquigarrow a \in R$ is used then $w \in \Sigma^*$. Consequently, $w = h(w) \in L(w_0)$ and $L(w_0)$ contains only square-free words.

Assume that the rule from R' used in $w \rightsquigarrow_{R'} w_1$ is among the rules (B.3)–(B.6) corresponding to a rule $x \rightsquigarrow a_1 \dots a_n$ in R.

If (B.3) is used then $w = uu''v$ and $w_1 = uu''a''_iv$ for $u, v \in \Sigma^*$ and $u'' \in (\Sigma'')^*$. Since u'' is a subword of w_1, it follows that u'' is square free. If $u'' \neq \varepsilon$ then w is square free, because otherwise u or v would contain a square, which contradicts w_1 being square free. If $w = uv$ then $w = h(w) \in L(w_0)$ and so it is square free.

If (B.4) is used then $w = uxa''_2 \dots a''_nv$ and $w_1 = ua'_1a''_2 \dots a''_nv$ for some $u, v \in \Sigma^*$. Since w_1 is square free, the same holds for u and v. If w contained a square then it would have to be a subword of ux. Since $h(w) = h_2(w) = uxv \in L(w_0)$, uxv is square free. Thus ux is square free as well, a contradiction.

If (B.5) or (B.6) is used then w contains a letter from Σ'. Thus $h(w) = h_1(w)$. Suppose that $w = uzzv$ for some $u, z, v \in (\Sigma \cup \Sigma' \cup \Sigma'')^*$. Hence

$$h(w) = h_1(uzzv) = h_1(u)h_1(z)h_1(z)h_1(v) \in L(w_0).$$

Since $L(w_0)$ contains only square-free words, we have $h_1(z) = \varepsilon$. Thus $z = \varepsilon$, since $h_1^{-1}(\varepsilon) = \{\varepsilon\}$. Consequently, $w = uzzv = uv$. □

It is easy to see that the conditional languages Σ^*, $\Sigma^*(\Sigma'')^*$, and $\Sigma^*\Sigma'(\Sigma'')^*$ are closed under the contraction rule. Also the last conditional language $a''_2 \dots a''_n\Sigma^*$ is closed under the contraction rule because the right-hand sides of all rules in R are square free (see Theorem B.3.1). Summarizing, we have the following theorem.

Theorem B.3.5. *There is an atomic CSRS $\langle \Sigma, R \rangle$ and $w_0 \in \Sigma^+$ such that the corresponding language $L(w_0)$ is undecidable and consists only of square-free words. Moreover, the conditional regular languages are closed under the contraction rule.*

B.4 Atomic conditional SRSs and $\mathcal{RL}_c$

In this section we show how to encode an atomic CSRS into the equational theory of $\mathcal{RL}_c$. Let $\langle \Sigma, R \rangle$ be the atomic CSRS from Theorem B.3.5 and $w_0 \in \Sigma^+$ such that the language $L(w_0) = \{ w \in \Sigma^+ \mid w \to^* w_0 \}$ consists only of square-free words (i.e. it is closed under the contraction rule) and is undecidable. Also conditional languages of every rule in R are closed under the contraction rule.

First, we describe the conditional contexts in our atomic CSRS by a right-linear context-free grammar. We can index the members of R by an index set I, i.e. $R = \{ R_i \mid i \in I \text{ and } R_i \text{ is a rule} \}$. Define an extended alphabet $\Sigma_e = \Sigma \cup \{ r_i \mid i \in I \}$, where r_i are fresh variables. Of course, we assume that $\Sigma_e \subseteq \mathit{Var}$. For every rule $R_i = \langle x \rightsquigarrow a, L_\ell, L_r \rangle$, where $x \in \Sigma^*$ and $a \in \Sigma$, define a regular language $L_i = L_\ell r_i L_r$. Note that the languages L_i are pairwise disjoint due to the pairwise different symbols r_i and closed under the contraction rule. Finally, we define a regular language $\mathsf{Aux} = \bigcup_{i \in I} L_i$, which is clearly closed under the contraction rule. Since Aux is regular, there is a right-linear context-free grammar G generating Aux. Consider the formula $\delta_G = 1 \wedge \bigwedge \Delta_G$ such that $w\delta_G \leq S$ holds in $\mathcal{RL}_c$ iff w belongs to Aux (see Theorem B.2.8), which means $w = ur_iv$ for some $i \in I$, $R_i = \langle x \rightsquigarrow a, L_\ell, L_r \rangle$, $u \in L_\ell$, and $v \in L_r$.

Second, we can combine this with the naïve way of encoding rules from Section B.3.1. As we have an atomic CSRS, which means only letters can occur on the right-hand side of rewriting rules, the main obstacle disappeared, cf. Section B.3.1. Moreover, we have shown how to describe the conditional contexts using the grammar G and hence δ_G. Now we can modify the definition of θ from Section B.3.1 in the following way.

For every rule $R_i = \langle x \rightsquigarrow a, L_\ell, L_r \rangle$ in R, we define a formula $\theta_i = x \backslash (a \vee r_i)$. Furthermore, we extend this for all the rules by defining a formula $\theta = 1 \wedge \bigwedge_{i \in I} \theta_i$. Note that $\theta \leq \theta_i$ for all $i \in I$ and $\theta \leq 1$. Hence $\theta \leq \theta^2 \leq 1 \cdot \theta = \theta$.

Assume we have $uxv, uav \in \Sigma^*$ and $R_i = \langle x \rightsquigarrow a, L_\ell, L_r \rangle \in R$. It is now impossible to show $ux\theta v \leq ua\theta v$ as in Section B.3.1, because θ_i contains r_i. This is by purpose—the conditionality of rewriting must be taken into account. Namely, ur_iv belongs to Aux only if $u \in L_\ell$ and $v \in L_r$. This gives us

$$ur_i\theta v\delta_G \leq ur_iv\delta_G \leq S \leq ua\theta v \vee S,$$

where $ur_iv\delta_G \leq S$ certifies that we rewrite in the correct context. Moreover, using $\delta_G \leq 1$ we know

$$ua\theta v\delta_G \leq ua\theta v \leq ua\theta v \vee S.$$

Hence we can combine these two things together using join and distributivity from Fact B.2.1 and so

$$u(a \vee r_i)\theta v\delta_G = ua\theta v\delta_G \vee ur_i\theta v\delta_G \leq ua\theta v \vee S.$$

We are now in a position to simulate the rewriting step $x \rightsquigarrow a$ of our atomic CSRS using the formula $\theta_i = x\backslash(a \vee r_i)$. Since $ux(x\backslash(a \vee r_i))\theta v\delta_G \leq u(a \vee r_i)\theta v\delta_G$ by Fact B.2.1, we obtain

$$\begin{aligned} ux\theta v\delta_G \leq ux\theta^2 v\delta_G \leq ux\theta_i\theta v\delta_G = \\ = ux(x\backslash(a \vee r_i))\theta v\delta_G \leq u(a \vee r_i)\theta v\delta_G \leq ua\theta v \vee S. \end{aligned}$$

It is clear that we need the formula θ to be spread everywhere along a word if we want to simulate an arbitrary rewriting step. For this reason, given a non-empty word $w = a_1 \dots a_n$ such that all a_i are letters, we define $w^\theta = a_1\theta a_2\theta \dots a_n\theta$.[4] Observe that $w^\theta \leq w$, $w^\theta \leq w^\theta\theta$, and $(uv)^\theta = u^\theta v^\theta$ hold for all non-empty words u, v, and w.

The following lemma shows in full details that the outlined construction can be used to describe atomic CSRSs in the language of $\mathcal{RL}_c$.

Lemma B.4.1 (Soundness)**.** *Let $w \in L(w_0)$. Then $w^\theta\delta_G \leq w_0 \vee S$.*

Proof. By induction on the length of derivation. The base case is obvious, since $w_0^\theta \leq w_0$ and $\delta_G \leq 1$. Hence $w_0^\theta\delta_G \leq w_0 \leq w_0 \vee S$. Assume that $w \to w_1$ by a rule $R_i = \langle x \rightsquigarrow a, L_\ell, L_r\rangle$ and $w_1 \to^* w_0$. By the induction hypothesis, we have $w_1^\theta\delta_G \leq w_0 \vee S$. Furthermore, we have $w = uxv$ and $w_1 = uav$ such that $u \in L_\ell$ and $v \in L_r$. We know that

$$\begin{aligned} w^\theta = u^\theta x^\theta v^\theta \leq u^\theta x^\theta\theta v^\theta \leq u^\theta x\theta v^\theta \leq u^\theta x\theta^2 v^\theta \leq u^\theta x(x\backslash(a \vee r_i))\theta v^\theta \leq \\ \leq u^\theta(a \vee r_i)\theta v^\theta = u^\theta a\theta v^\theta \vee u^\theta r_i\theta v^\theta = (uav)^\theta \vee (ur_iv)^\theta \leq w_1^\theta \vee ur_iv. \end{aligned}$$

Observe that $ur_iv \in L_i$. Thus $ur_iv\delta_G \leq S$ by Theorem B.2.8. Consequently,

$$w^\theta\delta_G \leq (w_1^\theta \vee ur_iv)\delta_G = w_1^\theta\delta_G \vee ur_iv\delta_G \leq w_0 \vee S.$$

□

[4]Note that for our atomic CSRS $\langle\Sigma, R\rangle$ from Theorem B.3.5 there is no need to define ε^θ because ε never occurs if we start rewriting from a non-empty word $w \in \Sigma^+$. In general, this need not be the case, but even then we could ensure such behaviour by expressing $w \to^* w_0$ equivalently as $bw \to^* bw_0$, where b is a fresh letter.

It remains to prove the opposite direction. Consider the residuated frame $\mathbf{W}_{L(w_0)\cup\mathsf{Aux}} = \langle \Sigma_e^*, \Sigma_e^* \times \Sigma_e^*, N\rangle$ where

$$z\ N\ \langle u, v\rangle \quad \text{iff} \quad uzv \in L(w_0) \cup \mathsf{Aux}.$$

Hence $\mathbf{W}^+_{L(w_0)\cup\mathsf{Aux}}$ is a residuated lattice. Moreover, $\mathbf{W}^+_{L(w_0)\cup\mathsf{Aux}} \in \mathcal{RL}_c$ by Lemma B.2.5 because $L(w_0) \cup \mathsf{Aux}$ is closed under the contraction rule by Theorem B.3.5.

Assume that $w^\theta\delta_G \le w_0 \vee S$ holds in $\mathcal{RL}_c$ for some $w \in \Sigma^+$. We want to show that $w \in L(w_0)$. Since the inequality $w^\theta\delta_G \le w_0 \vee S$ holds, we have $e(w^\theta\delta_G) \subseteq e(w_0 \vee S)$ for every $\mathbf{W}^+_{L(w_0)\cup\mathsf{Aux}}$-evaluation $e\colon Fm \to W^+_{L(w_0)\cup\mathsf{Aux}}$. Let e be the evaluation defined in Lemma B.2.7, i.e. $e(a) = \gamma_N\{a\}$ for every $a \in \Sigma_e$ and $e(A) = \gamma_N(L(A))$ for every non-terminal from the grammar G generating the language $\mathsf{Aux} = L(S)$. Therefore $e(u) = \gamma_N\{u\}$ for every word $u \in \Sigma^*$ by Lemma B.2.4. Furthermore, we have

$$e(w_0 \vee S) = \gamma_N(\gamma_N\{w_0\} \cup \gamma_N(L(S))) = \gamma_N(\{w_0\} \cup \mathsf{Aux}).$$

Since $\langle \varepsilon, \varepsilon\rangle \in (\{w_0\} \cup \mathsf{Aux})^\triangleright$, we have by Lemma B.2.3 that

$$\gamma_N(\{w_0\} \cup \mathsf{Aux}) \subseteq \{\langle \varepsilon, \varepsilon\rangle\}^\triangleleft = L(w_0) \cup \mathsf{Aux}.$$

Thus we know that the words in $e(w^\theta\delta_G) = \gamma_N(e(w^\theta) \cdot e(\delta_G))$ belong to $L(w_0) \cup \mathsf{Aux}$.

Lemma B.4.2. *It holds $\varepsilon \in e(\theta_i)$ for all $i \in I$. Consequently, $\varepsilon \in e(\theta) = \gamma_N\{\varepsilon\} \cap \bigcap_{i\in I} e(\theta_i)$.*

Proof. Consider the formula $\theta_i = x\backslash(a \vee r_i)$ corresponding to a rule $R_i = \langle x \rightsquigarrow a, L_\ell, L_r\rangle \in R$. It suffices to check that

$$\gamma_N\{x\} = e(x) \subseteq e(a \vee r_i) = \gamma_N(\gamma_N\{a\} \cup \gamma_N\{r_i\}) = \gamma_N\{a, r_i\}.$$

Using the basis for γ_N (see Lemma B.2.3) and (B.2), we have to show that $\{a, r_i\} \subseteq \{\langle u, v\rangle\}^\triangleleft$ implies $x \in \{\langle u, v\rangle\}^\triangleleft$. Assume that $uav, ur_iv \in L(w_0) \cup \mathsf{Aux}$. Hence $uav \in L(w_0)$ and $ur_iv \in L_i \subseteq \mathsf{Aux}$ which yield $u \in L_\ell$ and $v \in L_r$. Consequently, $uxv \to uav \to^* w_0$ and so $uxv \in L(w_0)$, i.e. $x \in \{\langle u, v\rangle\}^\triangleleft$. □

Since $\varepsilon \in e(\theta)$, we have $w \in e(w^\theta)$. Similarly, $\varepsilon \in e(\delta_G)$ by Lemma B.2.7 and so $w \in e(w^\theta)e(\delta_G)$. Consequently, $w \in \gamma_N(e(w^\theta)e(\delta)) = e(w^\theta\delta_G) \subseteq L(w_0) \cup \mathsf{Aux}$. As $w \notin \mathsf{Aux}$ we have $w \in L(w_0)$.

Lemma B.4.3 (Completeness)**.** *Assume that $w^\theta\delta_G \le w_0 \vee S$ holds in $\mathcal{RL}_c$ for $w \in \Sigma^+$. Then $w \in L(w_0)$.*

Since the problem whether $w \in L(w_0)$ is undecidable, the set $\{\, \varphi \leq \psi \mid \varphi \leq \psi \text{ holds in } \mathcal{RL}_c \,\}$ is undecidable as well. Thus we obtain the following theorem.

Theorem B.4.4. *The equational theory of $\mathcal{RL}_c$ is undecidable. Consequently, the set of formulae provable in $\mathbf{FL}_\mathbf{c}^+$ is undecidable.*

B.5 Conclusions

Theorem B.4.4 implies also the undecidability of the logic $\mathbf{FL_c}$, since $\mathbf{FL}_\mathbf{c}^+$ is its positive fragment. Recall that $\mathbf{FL_c}$ is an expansion of $\mathbf{FL}_\mathbf{c}^+$ where the language is expanded by a constant 0 (see [Gal+07]). The sequent calculus for $\mathbf{FL_c}$ can be obtained from the one shown in Definition B.2.1 by adding the following axiom and rule:

$$(0\mathrm{L})\ \frac{}{0 \Rightarrow} \qquad\qquad (0\mathrm{R})\ \frac{\Gamma \Rightarrow}{\Gamma \Rightarrow 0}$$

Moreover, the notion of a sequent is modified a little bit by allowing on the right-hand side a stoup, i.e. a formula or the empty sequence. Accordingly, one has to modify the rules from Definition B.2.1 in an obvious way. The logic $\mathbf{FL_{co}}$ is an extension of $\mathbf{FL_c}$ by the right weakening rule:

$$(\mathrm{o})\ \frac{\Gamma \Rightarrow}{\Gamma \Rightarrow \varphi}$$

The logic $\mathbf{FL_c}$ is sound and complete with respect to the variety of pointed square-increasing residuated lattices (also called $\mathbf{FL_c}$-algebras). An $\mathbf{FL_c}$-algebra $\mathbf{A} = \langle A, \wedge, \vee, \cdot, \backslash, /, 0, 1\rangle$ is an algebra such that $\langle A, \wedge, \vee, \cdot, \backslash, /, 1\rangle \in \mathcal{RL}_c$ and $0 \in A$. Similarly, the logic $\mathbf{FL_{co}}$ is sound and complete with respect to a subvariety of $\mathbf{FL_c}$-algebras axiomatized by the identity $0 \leq x$.

The logics $\mathbf{FL_c}$ and $\mathbf{FL_{co}}$ have a common fragment, namely $\mathbf{FL}_\mathbf{c}^+$. This follows easily for $\mathbf{FL_c}$ because every square-increasing residuated lattice is a reduct of an $\mathbf{FL_c}$-algebra, it suffices to interpret 0 arbitrarily. Concerning $\mathbf{FL_{co}}$, it is sufficient to show that every square-increasing residuated lattice $\mathbf{A}$ is embeddable into an $\mathbf{FL_c}$-algebra where 0 is its bottom element. Since $\mathbf{A}$ need not have a minimum, we can first embed it into its Dedekind–MacNeille completion $\bar{\mathbf{A}}$. Since the Dedekind–MacNeille completion preserves the identity $x \leq x^2$, we have $\bar{\mathbf{A}} \in \mathcal{RL}_c$ (see [CGT12]). Consequently, $\bar{\mathbf{A}}$ forms an $\mathbf{FL_c}$-algebra where $0 \leq x$ holds if we interpret 0 as the bottom element, it exists as the lattice reduct of $\bar{\mathbf{A}}$ is complete.

Theorem B.5.1. *The set of formulae provable in $\mathbf{FL_c}$ (resp. $\mathbf{FL_{co}}$) is undecidable.*

B.5.1 Used language

In our constructions we have used almost complete language, but this is not necessary. The constant 1 has been used for simplicity and can be easily eliminated. We know that $x = 1\backslash x$ and in all the remaining cases (in δ_G and θ) we can replace 1 by the meet of all $p\backslash p$ for all atoms p occurring in our construction.

Similarly, we can get rid of all fusions. First, we can change even the original problem $w \to^* w_0$ from Theorem B.3.1 into an equivalent problem $w \to^* p$, where p is a fresh atom, by adding the rewriting rule $w_0 \to p$. Second, using the same construction as in the paper we obtain an identity. We easily get an equivalent identity that contains no fusion using $x_1 \cdot \dots \cdot x_n \leq y$ iff $x_n \leq x_{n-1}\backslash(\dots\backslash(x_1\backslash y)\dots)$ and $x_1 \cdot \dots \cdot x_m\backslash y = x_m\backslash(\dots\backslash(x_1\backslash y)\dots)$. Notice that in our case such a y can only be an atom or join of atoms.

It should be also noted that we could use / instead of \ changing the construction accordingly. Therefore, we can clearly formulate the whole construction in the language containing only an implication, join, and meet. It does not matter whether as an identity in $\mathcal{RL}_c$ or sequent in $\mathbf{FL}_\mathbf{c}^+$ (with or without empty left-hand side).

B.5.2 Knotted axioms

It should be clear that the construction can be easily adapted for logics having a weaker form of contraction $x^k \leq x^l$, $1 \leq k < l$. Basically one has to change the encoding by replacing w^θ with w^{θ^k}, where if $w = a_1 \dots a_n$ then $w^{\theta^k} = a_1\theta^k \dots a_n\theta^k$. Furthermore, the final inequality is changed to $w^{\theta^k}\delta_G^k \leq w_0 \vee S$.

In order to modify our proof, note that the identity $x \leq x^2$ is used only for θ and δ_G. Since $\theta \leq 1$ and $\delta_G \leq 1$, we obtain $\theta^k = \theta^{k+1}$ and $\delta_G^k = \delta_G^{k+1}$. If 1 is not in the language and we change θ and δ_G according to Section B.5.1 then we still have $a\theta^k = a\theta^{k+1}$ and $a\delta_G^k = a\delta_G^{k+1}$ for every atom a occurring in $w^{\theta^k}\delta_G^k \leq w_0 \vee S$, which is sufficient to complete the proof.

B.5.3 Deduction theorem

From some point of view, one can understand our construction as a form of deduction theorem for a very limited fragment of formulae—a reachability problem for some rewriting systems is translated into provability in $\mathbf{FL}_\mathbf{c}^+$. However, this suffices to get a full form of "algorithmic" deduction theorem, because we can easily obtain the following chain of reductions. Let φ be a formula and T a finite theory. First, the set of formulae provable in $\mathbf{FL}_\mathbf{c}$

from T is recursively enumerable and hence there is a Minsky machine accepting an input (a suitable encoding of φ and T) iff φ is provable in $\mathbf{FL_c}$ from T. Second, this paper describes how to express such a decision problem in terms of provability in $\mathbf{FL_c^+}$. Moreover, all the steps in this chain are constructive and explicit.

Theorem B.5.2. *Let $T \cup \{\varphi\}$ be a finite set of formulae. There is an algorithm that produces a formula ψ (given an input φ and T) such that ψ is provable in $\mathbf{FL_c^+}$ iff φ is provable in $\mathbf{FL_c}$ from T.*

Acknowledgment

The authors wish to thank Wojciech Buszkowski and anonymous referees for valuable comments and remarks. The work was supported by the grant P202/11/1632 of the Czech Science Foundation and the long-term strategic development financing of the Institute of Computer Science (RVO:67985807).

Bibliography

[BO93] Ronald V. Book and Friedrich Otto. *String-rewriting Systems*. New York: Springer, 1993.

[CGT12] Agata Ciabattoni, Nikolaos Galatos and Kazushige Terui. 'Algebraic proof theory for substructural logics: Cut-elimination and completions'. *Annals of Pure and Applied Logic* 163.3 (Mar. 2012), pp. 266–290. DOI: 10.1016/j.apal.2011.09.003.

[Cho79] Laurent Chottin. 'Strict deterministic languages and controlled rewriting systems'. In: *Automata, Languages and Programming*. Ed. by Hermann A. Maurer. Vol. 71. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1979, pp. 104–117. ISBN: 978-3-540-09510-1. DOI: 10.1007/3-540-09510-1_9.

[Chvar] Karel Chvalovský. 'Undecidability of Consequence Relation in Full Non-associative Lambek Calculus'. *Journal of Symbolic Logic* (to appear).

[DP02] Brian A. Davey and Hilary A. Priestley. *Introduction to Lattices and Order*. 2nd. Cambridge: Cambridge University Press, 2002.

[Fre80] Ralph Freese. 'Free Modular Lattices'. *Transactions of the AMS* 261.1 (1980), pp. 81–91. DOI: 10.1090/S0002-9947-1980-0576864-X.

[Gal+07] Nikolaos Galatos, Peter Jipsen, Tomasz Kowalski and Hiroakira Ono. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*. Vol. 151. Studies in Logic and the Foundations of Mathematics. Amsterdam: Elsevier, Apr. 2007, p. 532.

[Gen35a] Gerhard Gentzen. 'Untersuchungen über das logische Schließen I'. *Mathematische Zeitschrift* 39.1 (1935), pp. 176–210. DOI: 10.1007/BF01201353.

[Gen35b] Gerhard Gentzen. 'Untersuchungen über das logische Schließen II'. *Mathematische Zeitschrift* 39.1 (1935), pp. 405–431. DOI: 10.1007/BF01201363.

[GJ13] Nikolaos Galatos and Peter Jipsen. 'Residuated frames with applications to decidability'. *Transactions of the AMS* 365 (2013), pp. 1219–1249. DOI: 10.1090/S0002-9947-2012-05573-5.

[Hor15] Rostislav Horčík. 'Word Problem for Knotted Residuated Lattices'. *Journal of Pure and Applied Algebra* 219.5 (May 2015), pp. 1548–1563. DOI: 10.1016/j.jpaa.2014.06.015.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* 1st. Reading, Massachusetts: Addison-Wesley, 1979.

[JT02] Peter Jipsen and Constantine Tsinakis. 'A Survey of Residuated Lattices'. In: *Ordered Algebraic Structures:* ed. by Jorge Martínez. Vol. 7. Developments in Mathematics. Dordrecht: Kluwer, 2002.

[Kan95] Max I. Kanovich. 'The Direct Simulation of Minsky Machines in Linear Logic'. In: *Advances in Linear Logic.* Ed. by Jean-Yves Girard, Yves Lafont and Laurent Regnier. London Mathematical Society Lecture Note Series 222. Cambridge University Press, 1995.

[KO91] Eiji Kiriyama and Hiroakira Ono. 'The contraction rule and decision problems for logics without structural rules'. *Studia Logica* 50.2 (1991), pp. 299–319. ISSN: 0039-3215. DOI: 10.1007/BF00370189.

[Kom86] Yuichi Komori. 'Predicate logics without the structural rules'. *Studia Logica* 45.4 (1986), pp. 393–404. ISSN: 0039-3215. DOI: 10.1007/BF00370272.

[Kri59] Saul Aaron Kripke. 'The problem of entailment'. *Journal of Symbolic Logic* 24 (1959). abstract, p. 324. URL: http://www.jstor.org/stable/2963903.

[KS95] Olga G. Kharlampovich and Mark V. Sapir. 'Algorithmic Problems In Varieties'. *International Journal of Algebra and Computation* 5 (1995), pp. 379–602.

[Laf96] Yves Lafont. 'The undecidability of second order linear logic without exponentials'. *Journal of Symbolic Logic* 61.2 (1996), pp. 541–548. DOI: 10.2307/2275674.

[Lin+92] Patrick Lincoln, John Mitchell, Andre Scedrov and Natarajan Shankar. 'Decision problems for propositional linear logic'. *Annals of Pure and Applied Logic* 56.1–3 (1992), pp. 239–311. DOI: 10.1016/0168-0072(92)90075-B.

[Lot02] M. Lothaire. *Algebraic Combinatorics on Words*. Encyclopedia of Mathematics and its Applications 90. Cambridge: Cambridge University Press, Apr. 2002.

[SO96] Bayu Surarso and Hiroakira Ono. 'Cut elimination in noncommutative substructural logics'. *Reports on Mathematical Logic* 30 (1996), pp. 13–29.

[Urq84] Alasdair Urquhart. 'The Undecidability of Entailment and Relevant Implication'. *Journal of Symbolic Logic* 49.4 (1984), pp. 1059–1073. DOI: 10.2307/2274261.