On the Complexity of RSRL

Stephan Kepser[*]
SFB 441, University of Tübingen
Nauklerstr. 35, 72074 Tübingen, Germany
`kepser@sfs.uni-tuebingen.de`

Abstract

In this paper we present a computability and a complexity result on Relational Speciate Reentrant Logic (RSRL). RSRL is a description logic designed to formalise the linguistic framework and theory Head-Driven Phrase Structure Grammar. We show here that given an RSRL-formula and a finite RSRL-interpretation it is in general not decidable if the formula is true in the given interpretation by reduction to Post Correspondence Problems. For so-called chainless RSRL, a semantically weaker version in which the expressive power of RSRL is significantly reduced, we show that if a class of finite structures is definable in chainless RSRL it is decidable by a Turing machine polynomially time bounded in the size of the input structures.

1 Introduction

RELATIONAL SPECIATE REENTRANT LOGIC (henceforce RSRL, [11, 12]) is a description logic designed by Frank Richter, Manfred Sailer, and Gerald Penn to formalise HEAD-DRIVEN PHRASE STRUCTURE GRAMMAR [8, 9]. HPSG is one of the leading paradigms in current linguistic research. It is in particular characterised by a high degree of formalisation and in opposite to its contender GB non-transformational. RSRL is based on SPECIATE REENTRANT LOGIC [5, 6], a feature logic for HPSG developed by King. RSRL, as SRL, is a description logic, it is designed to describe (linguistic) entities and not to talk about truth. A description of this logic can be true or false of an object. Therefore the denotation of such a description are those objects on which the description fits or for which the description is true. A linguistic entity is described mainly by sorts and features. Sorts describe the type of the entity, such as *word* or *phrase* or *subcat*; they act as unary relations partitioning the domain of discourse. Features describe sub-parts of entities such as the PHON (phonetic/phonologic) or SYNSEM (syntactic and semantic) sub-parts of a *word*. The main two extensions of RSRL over SRL is the introduction of arbitrary relations and quantification. Relations and quantification are specific to RSRL and not classical. Richter [11] argues at length that these extensions are required to fully formalise the principles of HPSG [9]. Here, we will not enter into the linguistic discussion. Neither can we explain the design

[*]This research is supported by a DFG-grant (SFB 441-01).

goals of RSRL. We will rather investigate computational and complexity properties of RSRL. In doing so we want to contribute to a better understanding of this feature logic, that is by some people regarded as *the* logic for HPSG.

The notion of the complexity of a logic stems from descriptive complexity theory. In principle, classes of finite structures can be defined in (at least) two different fashions. Classically, classes of structures are defined by logical axiomatisations. Hence it is the expressive power of the logic that decides which classes can be defined. Computationally, classes of structures can be defined by means of the computing devices that can compute if a candidate structure belongs to the class or if it does not. Hence it is the computational complexity that decides which classes can be defined. Descriptive complexity theory bridges between these two approaches by investigating the relationship between the expressive power of a logic on the one side and the complexity of a computing machine on the other. So, there are two questions: Given classes of structures defined in a particular logic, what is the required computational power to decide these classes? An example answer is that classes of structures defined in classical first order logic plus transitive closure require LOGSPACE-bounded non-deterministic Turing machines. And given classes of structures decided by some computing machine, what is the logic that can define these classes? Again an example answer is that classes of ordered structures that can be decided by LOGSPACE-bounded non-deterministic Turing machines can be axiomatised by first order logic plus transitive closure. Thus on finite ordered structures first order logic with transitive closure and LOGSPACE-bounded non-deterministic Turing machines define exactly the same classes of structures, as Immerman showed [3].

In this paper, we start from classes of finite structures definable in RSRL and search for the computing device that is required to decide these classes. To do so, we will systematically distinguish between two types of RSRL: RSRL as defined by Richter and so-called chainless RSRL, which is a semantically weaker version of RSRL. Richter introduces the notion of a chain. A chain is a potentially empty, finite sequence of objects in the denotation domain. One of the peculiarities of RSRL is the fact that the denotation of a variable may not only be a single object, but also a chain of objects. In chainless RSRL, a variable is always assigned to a single object. So, the syntax of RSRL and chainless RSRL are identical, while the semantics differs in that the denotation domain of a variable in chainless RSRL is always a single object of the universe (as is the case in classical logic) while it can be a single object or a chain of objects in (general) RSRL. Unsurprisingly, the introduction of chains has quite an impact on the expressive power and computability of RSRL, as we will show. Richter [11] argues that they are necessary to formalise the meaning of HPSG-principles that contain references to sets or lists like, e.g., the SUBCATEGORIZATION Principle.

After a short formal introduction to RSRL in the second section we present the first main result in the third one by showing that truth of a formula in a finite RSRL-interpretation is undecidable. Kepser showed in [4] that satisfiability of an SRL-formula is decidable, while King and Simov [7] proved the stronger notion of grammaticality of an SRL-formula to be undecidable. It is well known that satisfiability of classical first order logic is undecidable [1]. On the other hand, first order logic plus transitive closure is in LOGSPACE, as Immerman [3] showed. That is to say, given a formula of first-order logic with transitive closure and a finite first order structure, the complexity to calculate whether the formula holds true in the given structure is LOGSPACE bounded by the size of the structure. It turns out that the corresponding question for RSRL, namely given an RSRL-formula and a finite RSRL-structure, is the formula true in the structure is in general undecidable. The undecidability is proven by showing that Post Correspondence Problems can be coded in RSRL using chains. This result is quite remarkable since it shows that the seemingly innocent addition of chains is really a one with far reaching consequence. Checking the truth or falsehood of a formula in a *finite* structure is amongst the minimal requirements one can put onto a logic. Also, King and Simov's undecidability proof needs infinite structures. We remain in the realm of finite structures, and the introduction of chains, which are also finite, is motivated by linguistic needs.

The fourth section shows that chainless RSRL is in PTIME. After we saw the impact of allowing chains it is natural to ask where does RSRL stand if chains were removed. One can show that the classes of finite structures definable in chainless RSRL are decidable by PTIME-bounded Turing machines. To state that differently, given a Turing machine that represents a chainless RSRL formula and a finite RSRL-structure as input to the machine, the denotation of the formula in that structure can be computed in a time polynomial to the size of the structure. Or taken chainless RSRL as a query language, the complexity class of evaluating a query in a finite structure is PTIME. This result is a lot closer to the classical result for first order logic. That we still end up in a different complexity class is a consequence of the fact that RSRL is a description logic, as we will argue below.

2 RSRL

RSRL is a logic designed to describe HPSG-feature structures. An HPSG feature structure is a directed rooted finite hyper graph of a particular kind with sorts as node labels and features as edge labels. Features are functional. The distribution of sorts and features is not arbitrary, there are strong co-occurrence restrictions that allow the appearance of certain features only in the context of certain sorts. An example of how an HPSG feature structure looks like is given in Figure 1. It is the structure of the English pronoun *she* ([9], p. 17).

A full introduction to RSRL in its linguistic motivations or design decisions is far beyond the scope of this paper. This section rather summarises the technical definitions of RSRL that define its core. It is an excerpt of Chapter 3.1.1 "The Description Language" of [11] where a rich explanation with examples of all the technical definitions that follow can be found.

Definition 1 Σ is a ***signature*** iff

Σ is a septuple $\langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$,
$\langle \mathcal{G}, \sqsubseteq \rangle$ is a partial order of sorts,
$\mathcal{S} = \left\{ \sigma \in \mathcal{G} \,\middle|\, \begin{array}{l} \text{for each } \sigma' \in \mathcal{G}, \\ \text{if } \sigma' \sqsubseteq \sigma \text{ then } \sigma = \sigma' \end{array} \right\}$, the set of most specific sorts
$\mathcal{A}$ is a set of attributes or features,

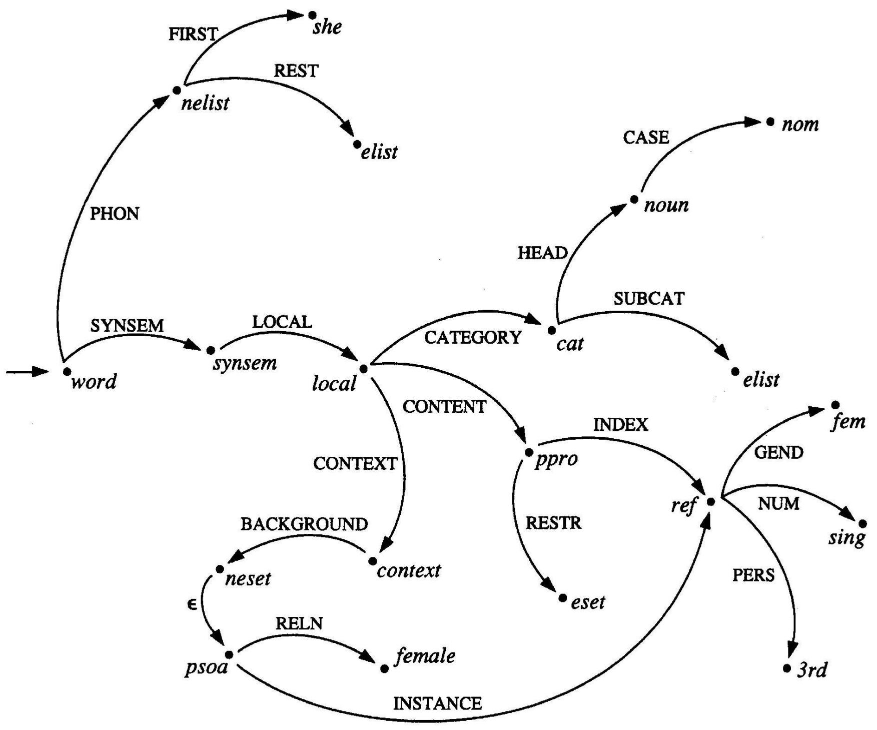

Figure 1: Structure of the English pronoun *she*

> $\mathcal{F}$ is a partial function from the Cartesian product of $\mathcal{G}$ and $\mathcal{A}$ to $\mathcal{G}$,
> for each $\sigma_1 \in \mathcal{G}$, for each $\sigma_2 \in \mathcal{G}$ and for each $\alpha \in \mathcal{A}$,
>
> > if $\mathcal{F}\langle\sigma_1,\alpha\rangle$ is defined and $\sigma_2 \sqsubseteq \sigma_1$
> > then $\mathcal{F}\langle\sigma_2,\alpha\rangle$ is defined and $\mathcal{F}\langle\sigma_2,\alpha\rangle \sqsubseteq \mathcal{F}\langle\sigma_1,\alpha\rangle$,
>
> $\mathcal{R}$ is a finite set of relation symbols, and
> $\mathcal{AR}$ is a total function from $\mathcal{R}$ to $\mathbb{N}^+$, the arities of the relations.

The partial order $\langle\mathcal{G},\sqsubseteq\rangle$ of sorts exists mainly for linguistic purposes. The technically relevant part of it is the set $\mathcal{S}$ of its most specific sorts. Therefore, other than in this section, the partial order will remain unmentioned in our work. $\mathcal{F}$ defines compatibility restrictions between sorts and features. It enables the linguist to state that only certain sorts and features may co-occur.

Suppose M is a set. M^* is the set of finite strings over M, including the empty string. We will refer to it as the set of finite sequences of elements of M. Similarly, M^+ is the set of nonempty finite sequences of elements of M. For convenience, we will henceforth write $\overline{M}$ as an abbreviation for $M \uplus M^*$.

RSRL signatures are interpreted as follows:

Definition 2 For each signature $\Sigma = \langle\mathcal{G},\sqsubseteq,\mathcal{S},\mathcal{A},\mathcal{F},\mathcal{R},\mathcal{AR}\rangle$, I is a Σ ***interpretation*** iff

I is a quadruple $\langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$,
U is a set, the universe or carrier,
S is a total function from U to $\mathcal{S}$,
A is a total function from $\mathcal{A}$ to the set of partial functions from U to U,
for each $\alpha \in \mathcal{A}$ and each $u \in \mathsf{U}$,

> if $\mathsf{A}(\alpha)(u)$ is defined
> then $\mathcal{F}\langle \mathsf{S}(u), \alpha \rangle$ is defined, and $\mathsf{S}(\mathsf{A}(\alpha)(u)) \sqsubseteq \mathcal{F}\langle \mathsf{S}(u), \alpha \rangle$, and

for each $\alpha \in \mathcal{A}$ and each $u \in \mathsf{U}$,

> if $\mathcal{F}\langle \mathsf{S}(u), \alpha \rangle$ is defined then $\mathsf{A}(\alpha)(u)$ is defined,

R is a total function from $\mathcal{R}$ to the power set of $\bigcup_{n \in \mathbb{N}} \overline{\mathsf{U}}^n$, and

> for each $\rho \in \mathcal{R}$, $\mathsf{R}(\rho) \subseteq \overline{\mathsf{U}}^{\mathcal{AR}(\rho)}$.

We sometimes call interpretations also structures. As one can see in the last line, relations do not only range over tuples of individuals, but over tuples of chains of individuals.

Definition 3 For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$,

$$\widehat{\mathcal{G}} = \mathcal{G} \cup \{chain, echain, nechain, metatop\},$$
$$\widehat{\sqsubseteq} = \sqsubseteq \cup \{\langle echain, chain \rangle, \langle nechain, chain \rangle\} \cup \left\{ \langle \sigma, \sigma \rangle \,\middle|\, \sigma \in \widehat{\mathcal{G}} \backslash \mathcal{G} \right\} \cup \left\{ \langle \sigma, metatop \rangle \,\middle|\, \sigma \in \widehat{\mathcal{G}} \right\},$$
$$\widehat{\mathcal{S}} = \mathcal{S} \cup \{echain, nechain\}, \text{ and}$$
$$\widehat{\mathcal{A}} = \mathcal{A} \cup \{\dagger, \triangleright\}.$$

Definition 4 For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$,

$\widehat{\mathsf{S}}$ is the total function from $\overline{\mathsf{U}}$ to $\widehat{\mathcal{S}}$ such that

> for each $u \in \mathsf{U}$, $\widehat{\mathsf{S}}(u) = \mathsf{S}(u)$,
> for each $u_1 \in \mathsf{U}, \ldots,$ for each $u_n \in \mathsf{U}$,
> $$\widehat{\mathsf{S}}(\langle u_1, \ldots, u_n \rangle) = \begin{cases} echain & \text{if } n = 0, \\ nechain & \text{if } n > 0 \end{cases}, \text{ and}$$

$\widehat{\mathsf{A}}$ is the total function from $\widehat{\mathcal{A}}$ to the set of partial functions from $\overline{\mathsf{U}}$ to $\overline{\mathsf{U}}$ such that

> for each $\alpha \in \mathcal{A}$, $\widehat{\mathsf{A}}(\alpha) = \mathsf{A}(\alpha)$,
> $\widehat{\mathsf{A}}(\dagger)$ is the total function from U^+ to U such that for each $\langle u_0, \ldots, u_n \rangle \in \mathsf{U}^+$,
> $$\widehat{\mathsf{A}}(\dagger)(\langle u_0, \ldots, u_n \rangle) = u_0, \text{ and}$$
> $\widehat{\mathsf{A}}(\triangleright)$ is the total function from U^+ to U^* such that for each $\langle u_0, \ldots, u_n \rangle \in \mathsf{U}^+$,
> $$\widehat{\mathsf{A}}(\triangleright)(\langle u_0, \ldots, u_n \rangle) = \langle u_1, \ldots, u_n \rangle.$$

$\dagger$ returns the head (left-most element) of a non-empty chain, and $\triangleright$ its rest.

Let $\mathcal{VAR}$ be a countably infinite set of symbols, the variables.

Definition 5 For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, $\mathcal{T}^\Sigma$ is the smallest set such that

$: \in \mathcal{T}^\Sigma$,
for each $v \in \mathcal{VAR}$, $v \in \mathcal{T}^\Sigma$, and
for each $\alpha \in \widehat{\mathcal{A}}$ and each $\tau \in \mathcal{T}^\Sigma$, $\tau\alpha \in \mathcal{T}^\Sigma$.

We call each element of $\mathcal{T}^\Sigma$ a Σ term. A Σ term consists of either the reserved symbol ':' or a variable followed by a (possibly empty) string of symbols of the expanded attribute set. To determine the interpretation of a term, we need the notion of a variable assignment.

Definition 6 For each signature Σ, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$,

$Ass_{\mathsf{I}} = \overline{\mathsf{U}}^{\mathcal{VAR}}$ is the ***set of variable assignments in*** I.

We note that the denotation of a variable can be a chain of elements from U. Σ terms are interpreted as partial functions from U to $\overline{\mathsf{U}}$.

Definition 7 For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $ass \in Ass_{\mathsf{I}}$, T_{I}^{ass} is the total function from $\mathcal{T}^\Sigma$ to the set of partial functions from U to $\overline{\mathsf{U}}$ such that for each $u \in \mathsf{U}$,

$T_{\mathsf{I}}^{ass}(:)(u)$ is defined and $T_{\mathsf{I}}^{ass}(:)(u) = u$,
for each $v \in \mathcal{VAR}$, $T_{\mathsf{I}}^{ass}(v)(u)$ is defined and $T_{\mathsf{I}}^{ass}(v)(u) = ass(v)$,
for each $\tau \in \mathcal{T}^\Sigma$, for each $\alpha \in \widehat{\mathcal{A}}$,

> $T_{\mathsf{I}}^{ass}(\tau\alpha)(u)$ is defined
> iff $T_{\mathsf{I}}^{ass}(\tau)(u)$ is defined and $\widehat{\mathsf{A}}(\alpha)(T_{\mathsf{I}}^{ass}(\tau)(u))$ is defined, and
> if $T_{\mathsf{I}}^{ass}(\tau\alpha)(u)$ is defined
> then $T_{\mathsf{I}}^{ass}(\tau\alpha)(u) = \widehat{\mathsf{A}}(\alpha)(T_{\mathsf{I}}^{ass}(\tau)(u))$.

We now define the set of descriptions or formulae of RSRL.

Definition 8 For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, $\mathfrak{D}^\Sigma$ is the smallest set such that

for each $\sigma \in \widehat{\mathcal{G}}$, for each $\tau \in \mathcal{T}^\Sigma$, $\tau \sim \sigma \in \mathfrak{D}^\Sigma$,
for each $\tau_1 \in \mathcal{T}^\Sigma$, for each $\tau_2 \in \mathcal{T}^\Sigma$, $\tau_1 \approx \tau_2 \in \mathfrak{D}^\Sigma$,
for each $\rho \in \mathcal{R}$, for each $x_1 \in \mathcal{VAR}$, ..., for each $x_{\mathcal{AR}(\rho)} \in \mathcal{VAR}$,
  $\rho(x_1, \ldots, x_{\mathcal{AR}(\rho)}) \in \mathfrak{D}^\Sigma$,
for each $\delta \in \mathfrak{D}^\Sigma$, $\neg\delta \in \mathfrak{D}^\Sigma$,
for each $\delta_1 \in \mathfrak{D}^\Sigma$, for each $\delta_2 \in \mathfrak{D}^\Sigma$, $[\delta_1 \wedge \delta_2] \in \mathfrak{D}^\Sigma$,
for each $\delta_1 \in \mathfrak{D}^\Sigma$, for each $\delta_2 \in \mathfrak{D}^\Sigma$, $[\delta_1 \vee \delta_2] \in \mathfrak{D}^\Sigma$.
for each $x \in \mathcal{VAR}$, for each $\delta \in \mathfrak{D}^\Sigma$, $\exists x\, \delta \in \mathfrak{D}^\Sigma$,
for each $x \in \mathcal{VAR}$, for each $\delta \in \mathfrak{D}^\Sigma$, $\forall x\, \delta \in \mathfrak{D}^\Sigma$,

As usual, a formula without free variables is called a sentence. In RSRL, the quantification domains are so-called components of elements. Given an element in an interpretation, its components are all those elements that can be reached via some feature path:

Definition 9 For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, and for each $u \in \mathsf{U}$,

$$\mathsf{Co}_{\mathsf{I}}^{u} = \left\{ u' \in \mathsf{U} \;\middle|\; \begin{array}{l} \text{for some } ass \in Ass_{\mathsf{I}}, \\ \text{for some } \pi \in \mathcal{A}^*, \\ \quad T_{\mathsf{I}}^{ass}(:\pi)(u) \text{ is defined, and} \\ \quad u' = T_{\mathsf{I}}^{ass}(:\pi)(u) \end{array} \right\}.$$

The following definition is a variant of the usual definition of a modified variable assignment.

Definition 10 For each signature Σ, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $ass \in Ass_{\mathsf{I}}$, for each $v \in \mathcal{VAR}$, for each $w \in \mathcal{VAR}$, for each $u \in \overline{\mathsf{U}}$,

$$ass\tfrac{u}{v}(w) = \begin{cases} u & \text{if } v = w \\ ass(w) & \text{otherwise.} \end{cases}$$

Here is finally the definition of the denotation of a description.

Definition 11 For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $ass \in Ass_{\mathsf{I}}$, D_{I}^{ass} is the total function from $\mathfrak{D}^{\Sigma}$ to the power set of U such that

for each $\tau \in \mathcal{T}^{\Sigma}$, for each $\sigma \in \widehat{\mathcal{G}}$,

$$D_{\mathsf{I}}^{ass}(\tau \sim \sigma) = \left\{ u \in \mathsf{U} \;\middle|\; \begin{array}{l} T_{\mathsf{I}}^{ass}(\tau)(u) \text{ is defined, and} \\ \widehat{\mathsf{S}}(T_{\mathsf{I}}^{ass}(\tau)(u)) \mathrel{\widehat{\sqsubseteq}} \sigma \end{array} \right\},$$

for each $\tau_1 \in \mathcal{T}^{\Sigma}$, for each $\tau_2 \in \mathcal{T}^{\Sigma}$,

$$D_{\mathsf{I}}^{ass}(\tau_1 \approx \tau_2) = \left\{ u \in \mathsf{U} \;\middle|\; \begin{array}{l} T_{\mathsf{I}}^{ass}(\tau_1)(u) \text{ is defined,} \\ T_{\mathsf{I}}^{ass}(\tau_2)(u) \text{ is defined, and} \\ T_{\mathsf{I}}^{ass}(\tau_1)(u) = T_{\mathsf{I}}^{ass}(\tau_2)(u) \end{array} \right\},$$

for each $\rho \in \mathcal{R}$, for each $x_1 \in \mathcal{VAR}$, ..., for each $x_{\mathcal{AR}(\rho)} \in \mathcal{VAR}$,

$$\begin{array}{l} D_{\mathsf{I}}^{ass}(\rho(x_1, \ldots, x_{\mathcal{AR}(\rho)})) \\ = \left\{ u \in \mathsf{U} \;\middle|\; \langle ass(x_1), \ldots, ass(x_{\mathcal{AR}(\rho)}) \rangle \in \mathsf{R}(\rho) \right\}, \end{array}$$

for each $\delta \in \mathfrak{D}^{\Sigma}$,

$$D_{\mathsf{I}}^{ass}(\neg\delta) = \mathsf{U} \backslash D_{\mathsf{I}}^{ass}(\delta),$$

for each $\delta_1 \in \mathfrak{D}^{\Sigma}$, for each $\delta_2 \in \mathfrak{D}^{\Sigma}$,

$$D_{\mathsf{I}}^{ass}([\delta_1 \wedge \delta_2]) = D_{\mathsf{I}}^{ass}(\delta_1) \cap D_{\mathsf{I}}^{ass}(\delta_2),$$

for each $\delta_1 \in \mathfrak{D}^{\Sigma}$, for each $\delta_2 \in \mathfrak{D}^{\Sigma}$,

$$D_{\mathsf{I}}^{ass}([\delta_1 \vee \delta_2]) = D_{\mathsf{I}}^{ass}(\delta_1) \cup D_{\mathsf{I}}^{ass}(\delta_2).$$

for each $v \in \mathcal{VAR}$, for each $\delta \in \mathfrak{D}^{\Sigma}$,

$$D_{\mathsf{I}}^{ass}(\exists v\,\delta) = \left\{ u \in \mathsf{U} \,\middle|\, \begin{array}{l} \text{for some } u' \in \overline{\mathsf{Co}_{\mathsf{I}}^{u}}, \\ u \in D_{\mathsf{I}}^{ass\frac{u'}{v}}(\delta) \end{array} \right\},$$

for each $v \in \mathcal{VAR}$, for each $\delta \in \mathfrak{D}^{\Sigma}$,

$$D_{\mathsf{I}}^{ass}(\forall v\,\delta) = \left\{ u \in \mathsf{U} \,\middle|\, \begin{array}{l} \text{for each } u' \in \overline{\mathsf{Co}_{\mathsf{I}}^{u}}, \\ u \in D_{\mathsf{I}}^{ass\frac{u'}{v}}(\delta) \end{array} \right\}.$$

The term $\tau_1 \approx \tau_2$ originally expresses a path equation in a feature structure. In the special case where both τ_1 and τ_2 are variables, the term expresses equality of variables: The denotation of a variable is independent of the element of the carrier at which it is evaluated (Definition 7), so for variables $x, y \in \mathcal{VAR}$, $x \approx y$ is either true for every element of the carrier or false. And the definition of denotations above demands the two variables to denote the same thing. Thus RSRL provides a limited version of an equality logic.

It is in particular the interpretation of quantification which is special. It contains two unusual elements: localisation and transitive closure. Localisation because the quantification for a particular element in the universe ranges only over its components, and transitive closure, because the components form the transitive closure of the one step transition from one element in the universe to the next via features.

To get to the chainless variant of RSRL some of the above definitions need slight modifications. In Definition 2 of interpretations, relations must be relations on individuals, only, not on chains. The extended signature and its interpretation in Definitions 3 and 4 are no longer needed. And in Definition 6 variables can only be assigned to individuals, not chains.

3 Truth in a Finite RSRL-Structure is Undecidable

Since RSRL is a description logic, the notion of truth is only indirectly present. But already Richter observes that one can say an RSRL-formula φ is true in some interpretation I with variable assignment b if its denotation $D_{\mathsf{I}}^{b}(\varphi) = U_{\mathsf{I}}$ is the whole universe. In other words, a formula is true, if it is true for every object of the universe. In this case we just write $(\mathsf{I}, b) \models \varphi$. A formula is false, if it is not true.

One of the important differences between RSRL and classical logic is the fact that on the classical side the truth of a sentence in a given finite model can be decided while on the RSRL side it cannot in general. In the classical case there exists a result in descriptive complexity theory (see [2], Chapter 6) by Immerman [3] that shows that even if we enrich first order logic by deterministic transitive closure, the complexity for deciding whether or not a sentence is true in a given finite structure is in LOGSPACE in the size of the structure, so very low. The situation for RSRL is quite different.

Theorem 12 *Given a sentence φ and a finite RSRL-interpretation I, it is in general not decidable, if I models φ.*

We prove this theorem by coding Post correspondence problems in finite RSRL-structures. Let Γ be a finite alphabeth. A Post correspondence system [10] is a finite set P of ordered pairs of nonempty strings; that is P is a finite subset of $\Gamma^+ \times \Gamma^+$. Here is a simple example: Let $\Gamma = \{a,b\}$ and $P = \{(a,ab),(ba,a)\}$. A *match* of P is any string $w \in \Gamma^*$ such that, for some $n > 0$ and some (not necessarily distinct) pairs $(u_1,v_1),\ldots,(u_n,v_n) \in P$ it is the case that $w = u_1 \ldots u_n = v_1 \ldots v_n$. A match for the example is *aba*: $\substack{a \\ ab}\,\substack{ba \\ a}$ where $\substack{a \\ ab}$ is the first and $\substack{ba \\ a}$ the second pair. The Post correspondence problem is the question whether there exists a match for a given system P. Post showed that this question is in general undecidable (if Γ consists of more than 2 letters).

Let P be a Post correspondence system. It will be coded as follows. Each letter of Γ is a node in the feature structure, so Γ is the carrier of the RSRL-interpretation. There exists only one singe sort s, it is appropriate for all letters. The sort s does not serve any purpose in the coding. It is there only for the technical reason that each node of an interpretation must have a sort. There is a single feature f. Feature f is appropriate for sort s and sort s is appropriate for (s,f). Let k be the cardinality of Γ and e be an enumeration of Γ. We define $\langle e(i), e(i+1)\rangle \in f$ for each $0 < i < k$ and $\langle e(k), e(1)\rangle \in f$. Thus f forms a complete cycle around the letters. The order of the letters in the cycle is unimportant, the choice of the enumeration e hence free. All that is important is that for every pair of letters $u,v \in \Gamma$ there exists a path from u to v. Consequently, for each $u \in \Gamma$ the set of components $\mathsf{Co}^u = \Gamma$. This simplifies proofs significantly: Quantification now behaves (almost) classically. To continue the above example, here is its RSRL-coding:

$$\begin{tikzcd}
\boxed{a}_s \arrow[r, bend left, "f"] & \boxed{b}_s \arrow[l, bend left, "f"]
\end{tikzcd}$$

We define three relations, all of them are relations on *chains* of letters. The first relation, Post, contains all pairs of P, i.e., for all $u,v \in \Gamma^+ : (u,v) \in \mathsf{Post}$ iff $(u,v) \in P$. The second relation, Conc, defines concatenation of chains of letters:

$$\begin{aligned}
\forall xyz\, \mathsf{Conc}(x,y,z) \quad \leftrightarrow \quad & (\hat{S}(x) \sim \mathit{echain} \wedge y \approx z) \quad \vee \\
& (\hat{S}(x) \sim \mathit{nechain} \wedge \\
& \;\exists w r_1 r_2 : x\dagger \approx w \wedge z\dagger \approx w \wedge x\triangleright \approx r_1 \\
& \qquad \wedge z\triangleright \approx r_2 \wedge \mathsf{Conc}(r_1,y,r_2))
\end{aligned}$$

Lemma 13 *For all chains $x,y,z \in \Gamma^*$: $\mathsf{Conc}(x,y,z)$ if and only if z is the concatenation of x and y.*

Proof. Let $x,y,z \in \Gamma^*$ with $\mathsf{Conc}(x,y,z)$. We show that z is the concatenation of x and y by an induction on the length of x. If x is the empty chain, then the first disjunct holds and, since $y \approx z$ we have $y = z$. Clearly z is the concatenation of x and y. If x is non-empty, then, by the second disjunct, there is a letter w which is the head of x and the head of z and a chain r_1 which is the tail of x and a chain r_2 which is the tail of z such that $\mathsf{Conc}(r_1,y,r_2)$. Since the length of r_1 is smaller than that of x, it follows by the induction hypothesis that r_2 is the concatenation of r_1 and y. Thus wr_2 is the concatenation of wr_1 and y.

Let $x,y,z \in \Gamma^*$ with z being the concatenation of x and y. We show that $\mathsf{Conc}(x,y,z)$ by an induction on the length of x. If x is the empty chain, then $y = z$ and thus the first disjunct holds and therefore $\mathsf{Conc}(x,y,z)$. If x is non-empty, then we can split it up into the first letter w of x and the rest r_1 of x. Since z is the concatenation of x and y, we know we can split up z also and w must be the first letter of z; we call the rest r_2. Clearly, r_2 is the concatenation of r_1 and y. By induction hypothesis, $\mathsf{Conc}(r_1,y,r_2)$. Thus the second disjunct holds, and therefore $\mathsf{Conc}(x,y,z)$. ■

The third relation, Post-Chain, contains all pairs of chains that can be constructed by pairwise concatenating pairs from Post.

$$
\begin{aligned}
&\forall xy\ \mathsf{Post\text{-}Chain}(x,y) \quad \leftrightarrow \\
&\quad \mathsf{Post}(x,y) \quad \vee \\
&\quad (\exists u_1\, u_2\, v_1\, v_2\ \hat{S}(u_1) \sim \mathit{nechain} \wedge \hat{S}(u_2) \sim \mathit{nechain} \wedge \hat{S}(v_1) \sim \mathit{nechain} \\
&\qquad \wedge \hat{S}(v_2) \sim \mathit{nechain} \wedge \mathsf{Conc}(u_1,u_2,x) \wedge \mathsf{Conc}(v_1,v_2,y) \\
&\qquad \wedge \mathsf{Post\text{-}Chain}(u_1,v_1) \wedge \mathsf{Post}(u_2,v_2))
\end{aligned}
$$

A pair of chains is in Post-Chain iff it is a pair of the Post correspondence system Post or both components can be split up into subchains the first of which is in Post-Chain and the second a pair of the Post correspondence system Post.

Lemma 14 *For all chains $x,y \in \Gamma^+$: Post-Chain(x,y) if and only if there is an $n > 0$ and strings $u_1,\ldots,u_n,v_1,\ldots,v_n \in \Gamma^+$ such that $x = u_1 \ldots u_n$, $y = v_1 \ldots v_n$ and for each $1 \le i \le n : (u_i,v_i) \in P$.*

Proof. Let there be an $n > 0$ and strings $u_1,\ldots,u_n,v_1,\ldots,v_n \in \Gamma^+$ such that $x = u_1 \ldots u_n, y = v_1 \ldots v_n$ and for each $1 \le i \le n : (u_i,v_i) \in P$. We show Post-Chain$(x,y)$ by an induction on n.
Base case $n = 1$. In this case $(x,y) \in P$ and $(x,y) \in$ Post and therefore $(x,y) \in$ Post-Chain by the first disjunct of the definition.
Step case $n > 1$. Let $a_1 = u_1 \ldots u_{n-1}$ and $b_1 = v_1 \ldots v_{n-1}$. Then $(a_1,b_1) \in$ Post-Chain by induction hypothesis. a_1 and b_1 are non-empty chains by definition, and $x = a_1 u_n$ and $y = b_1 v_n$. Of course $(u_n,v_n) \in$ Post. Hence the second disjunct of the Post-Chain definition holds. And therefore $(x,y) \in$ Post-Chain.

Let $(x,y) \in$ Post-Chain. Then by definition of Post-Chain either $(x,y) \in P$ or x is the concatenation of the two non-empty chains u_1 and u_2, y is the concatenation of the two non-empty chains v_1, and v_2, $(u_2,v_2) \in P$ and $(u_1,v_1) \in$ Post-Chain, and u_1 is shorter than x as v_1 is shorter than y. The argument can be repeated for the pair (u_1,v_1) and so on. The splitting process must terminate since the resulting pair is always smaller. And each step chops off a pair from P. Hence (x,y) is indeed the concatenation of pair from P. ■

To continue our simple example from above, we use square brackets ([]) for list notation to show part of the denotation of Post, Conc, and Post-Chain:
$\mathsf{Post} = \{([a],[ab]),([ba],[a])\}$,
$\mathsf{Conc} \supset \{([a],[a],[aa]),([a],[b],[ab]),([a],[ab],[aab])\}$, and
$\mathsf{Post\text{-}Chain} \supset \{([a],[ab]),([ba],[a]),([aba],[aba]),([baa],[aab]),([aa],[abab])\}$.

Now consider the formula

$$\exists x\, \mathsf{Post\text{-}Chain}(x,x).$$

By the above lemma, this formula expresses that there is a match of the Post correspondence system. If truth of the conjunction of this formula together with the defining formulae of Conc and Post-Chain was decidable in the given RSRL-interpretation, we had a method for solving Post correspondence problems.

A simple consequence of Theorem 12 is that there cannot be a method that takes an arbitrary RSRL-formula and an RSRL-interpretation and computes the denotation of the formula in the interpretation.

Corollary 15 *Given an RSRL-formula* φ*, a variable assignment* b*, and a finite RSRL-interpretation* I*,* $D_I^b(\varphi)$*, the denotation of* φ *in* I *under* b*, is in general not computable.*

Compare these results with the already quoted one by Immerman [3] for classical first order logic. The key difference of course is the presence of chains as ranges for variables and argument positions of relations. Although we also need a logic that is expressive enough to state the definitions of Conc and Post-Chain, it is primarily the chains that we need to code Post correspondence problems. In some sense, this is a discussion on the notion of *finiteness* of an interpretation. An interpretation is finite, if its carrier, its set of nodes, is finite. But even then, the set of chains of these nodes is infinite. So, in some sense, we have an infinite domain here. One way of interpreting this is to say that RSRL can and perhaps should be seen as a two-sorted[1] logic. We have two related domains of denotation: a domain of nodes and a domain of finite sequences of nodes. When taking this view variables and relations must be regarded as inherently polymorphic. It seems unlikely that this polymorphism is really needed in the formalisation of HPSG. If it is indeed not, the above result could be taken as an argument to restate RSRL as a two-sorted logic, because a two-sorted reformulation would not only make the logic more perspicuous, it would also make the undecidability result look quite as one would have expected it.

4 Chainless RSRL is in PTIME

The uncomputability result of the previous section relies on the existence of chains. It is therefore natural to ask what complexity result can be obtained when leaving out chains. We can show that if a class of finite structures is definable in chainless RSRL, then it is decidable by a deterministic Turing machine in a time that is polynomial in the size of the input structures. In this section, we follow closely the book on finite model theory by Ebbinghaus and Flum [2], in particular Chapter 6. Let K be a class of finite RSRL-interpretations. We write $K \in$ RSRL if K is axiomatisable in RSRL. Axiomatisability in a description logic is to be read as follows. All those structures are axiomatised for which the defining formula is true, in other words has the whole carrier as denotation.

Let K be a class of finite interpretations and M a Turing machine. M accepts K if M accepts exactly those finite interpretations that lie in K. We define K is in PTIME

[1] *sorted*, of course, in the logical sense, not the HPSG sense.

(“deterministic polynomial time”) iff there exists a deterministic Turing machine M and a polynomial $p \in \mathbb{N}[x]$ such that M accepts K and M is p time-bounded.

We will now take RSRL-interpretations as inputs to Turing machines. We consider only finite interpretations, infinite ones cannot be input to a machine. Let I be a finite interpretation with $|I| = n$. By passing to an isomorphic copy we can and always will assume that the carrier $U_I = \{0, 1, \dots, n-1\}$, an initial sequence of the natural numbers.

The machine has input tapes and work tapes, all infinitely extending to the right. The input tapes are the universe tape, the sort tape, a feature tape for each feature and a relation tape for each relation. The universe tape contains just n consecutive 1’s, so it looks like this

α	1	1	$\dots$	1	0	0	$\dots$
-1	0	1		$n-1$	n	$n+1$	

The sort tape contains the sort for each element. The inscription of cell i is σ iff $S(i) = \sigma$ and the rest of the tape is filled with zeros.

Features and relations are both coded in the same way, features being special types of binary relations. To code relation R, let R be r-ary, that is, $R \subseteq \{0, \dots, n-1\}^r$. For $j < n^r$, let $|j|_r$ be the j-th r-tuple in the lexicographic ordering of $\{0, \dots, n-1\}^r$; in other words, look at the unique n-adic representation of j,

$$j = j_1 \cdot n^{r-1} + j_2 \cdot n^{r-2} + \dots + j_{r-1} \cdot n + j_r \text{ with } 0 \leq j_i < n,$$

and set $|j|_r = (j_1, \dots, j_r)$. Then the tape coding R has the inscription

α	a_0	a_1	a_2	a_3	$\dots$	a_{n^r-1}	0
-1	0	1	2	3		n^r-1	n^r

where $a_j = 1$ iff $R|j|_r$ and $a_j = 0$ iff not $R|j|_r$.

Formulae may contain free variables. The values of these variables are determined by assignment functions. So, in order to compute on structures we would need to code assignment functions, too. Instead, we look at a variant of RSRL which allows for individual constants. This obviously does not extend the expressive power of the language, since every constant can be replaced by a free variable. We will do just the opposite and replace every free variable by a *new* constant in an extended signature and fix the denotation of the constant to be exactly the element that is the denotation of the variable it replaces. This way, we need not code assignment functions. For every constant, there is an input tape. Since the denotation of a constant is a number $< n$, the constant is coded by the binary representation of that number.

The machine also possesses several work tapes. We define the first work tape to be the output tape. That is to say, if the machine halts, the first work tape contains the information which elements are in the denotation of the formula φ. So, the i-the cell of the tape contains a 1 iff $i \in D(\varphi)$ and a 0 otherwise. All other cells are padded with 0's.

We want to show that for any sentence of RSRL the class K of its finite models is in PTIME. We even show that there is a machine M strongly witnessing $K \in$ PTIME, that is,

- M accepts K;
- for any interpretation I every run of M, started with I stops; in particular, M decides K;
- for any interpretation I every run of M satisfies the polynomial time bound.

Theorem 16 *Let K be a class of finite interpretations of chainless RSRL. If $K \in$ RSRL then $K \in$ PTIME.*

Proof. The proof proceeds by induction on the axiomatising formula φ.
Let φ be atomic. It can have three different forms: $R(c_1, \dots, c_r)$ for some r-ary relation R; or $\tau_1 \approx \tau_2$ with terms τ_1, τ_2; or $\tau \sim \sigma$ with term τ and sort σ. In the first case, we compute the number l represented by $(c_1, \dots, c_r)$ in n-ary notation and then look up the l-the cell of the input tape for R. If it contains a 1, a 1 is written onto the first n cells of the first work tape. If it contains a 0, a 0 is written onto the first n cells of the first work tape. Since l is a polynome over n the result can be computed in polynomial time.

In the second case we have to compute the denotation of the terms τ_1, τ_2 for each element $0 \leq i < n$. Remember that by Definition 5 a term consists of the special symbol : or a constant followed by a possibly empty string of features. For each of the two terms we use a work tape containing the denotation of the term application onto i computed so far in binary notation. For the first term τ_1, the initial inscription of its work tape is i, iff τ_1 starts with : as leftmost symbol, or the value of the constant c_v as given from the input tape for that constant, iff c_v is the leftmost symbol. After initialisation, for each feature f we look up the result of applying f to the element j on the work tape. Even though f is represented as a relation, we find that value by checking if $(j,k) \in f$ for each $0 \leq k < n$. Since f is functional, for at most one k there is $(j,k) \in f$. If there is one, k is written onto the work tape and we proceed with the next feature of τ_1. If there is none, this means the term τ_1 is undefined on i. Thus we write a 0 onto the i's cell of the output tape and proceed with $i+1$. Analogously we calculate the application of τ_2 onto i. Finally we compare the values of the two work tapes. If they are the same number, we write a 1 onto the i's cell of the output tape. If the numbers are different, we inscribe a 0. Then we proceed with $i+1$.
Initialising a work tape is logarithmic in n. Application of a feature function is cubic in n. Hence the denotation of the formula can be computed in polynomial time.

The third case is somewhat similar to the second. For each $0 \leq i < n$, we calculate the denotation of term τ exactly as described in the case. If τ is undefined on i, we just note a 0 on the i's cell of the output tape. Otherwise let m be the result of applying τ to i. We check now the m's cell of the sort tape to see if it agrees with σ. If so, we write a 1 on the i's cell of the output tape, if not, we write a 0. Then we proceed with $i+1$.

Initialising a work tape is logarithmic in n. Application of a feature function is cubic in n. Checking a sort is linear in n. Hence the denotation of the formula can be computed in polynomial time.

Let φ be non-atomic. If $\varphi = \neg\psi$, then there is by induction hypothesis a machine that computes ψ in polynomial time. We take this machine and add a step at the end. In this step, the output on the first work tape is reversed by replacing on the first n cells every 1 by a 0 and vice versa. This step is clearly linear in n. Thus the machine for φ is in PTIME.

Let $\varphi = \psi \vee \chi$. By induction hypothesis there are PTIME-bounded machines M_ψ and M_χ for ψ and χ. A machine M_φ for φ is constructed by running M_ψ and copying the result to a work tape, running M_χ and copying the result onto another work tape and finally reading the results on the work tapes cell by cell and writing a 1 on the first work tape whenever there is a 1 on one of the two work tapes and a 0 otherwise, a step, which is apparently linear in n. Thus M_φ is in PTIME.

Let $\varphi = \psi \wedge \chi$. By induction hypothesis there are PTIME-bounded machines M_ψ and M_χ for ψ and χ. A machine M_φ for φ is constructed by running M_ψ and copying the result to a work tape, running M_χ and copying the result onto another work tape and finally reading the results on the work tapes cell by cell and writing a 1 on the first work tape whenever there is a 1 on both of the two work tapes and a 0 otherwise, a step, which is apparently linear in n. Thus M_φ is in PTIME.

Let $\varphi = \exists x\psi$. Remember that by Definition 11 $D(\exists x\psi) = \{0 \leq i < n \mid \exists j < n : j \in \mathsf{Co}^i$ and $i \in D(\psi^{j/x})\}$.[2] A machine M_φ for φ works in two steps. In a first step, it computes the components for every element of the carrier, a binary relation Components $Co \subseteq \{0,1,\ldots,n-1\}^2$ with $(i,j) \in Co$ iff $j \in \mathsf{Co}^i$. We give here a polynomial algorithm that can easily be used to define a Turing machine. The relation Co is initialised by setting $(i,i) \in Co$ for all $0 \leq i < n$. We need one additional boolean variable *new*.

```
Repeat
  Set new := false
  For i := 0 to n-1
    For j := 0 to n-1
      If Co(i,j) then
        For k := 0 to n-1
          For all features F
            If F(j,k) and not Co(i,k) then
              Set Co(i,k) := true
              Set new := true
Until not new
```

Since $Co \subseteq \{0,1,\ldots,n-1\}^2$ and Co grows monotone in each run of the repeat loop, the repeat loop terminates after at most n^2 steps. Therefore the complexity of the algorithm is $O(n^5)$. The incarnation as a machine is even worse, because looking up and writing to a cell of the representation of a binary relation – as are Co and the

[2] As mentioned afore, $\psi^{j/x}$ is gained from ψ by replacing every occurrence of the variable x by the new constant c_x and fixing the denotation of the constant c_x to j.

features – is quadratic in n. Hence the complexity for the machine is $O(n^7)$, bad but still polynomial, as desired.

It is of course clear that the Components relation needs to be computed only once, even if the formula contains several quantifiers.

The second step consists in calculating the denotation D of $\exists x\psi$. By induction hypothesis there is for every $0 \leq j < n$ a PTIME-machine $M_{\psi^{j/x}}$ for ψ where the variable x is replaced by the constant j. Again we provide a pseudo-code algorithm. D is initialised by setting everything to false, analog to erasing the first work tape.

```
For i := 0 to n-1
  Set j := 0
  Set found := false
  While (j < n and not found)
    If Co(i,j) then
      Run $M_{\psi^{j/x}}$
      If i $\in$ Output($M_{\psi^{j/x}}$) then
        Set $D$(i) := true
        Set found := true
    Set j := j+1
```

The algorithm has the complexity $O(n^2)$ multiplied by the complexity of M_ψ. Since M_ψ is in PTIME by induction hypothesis, it follows that M_φ is in PTIME, too.

Let $\varphi = \forall x\psi$. This case is analog to the existential quantification case. ■

As a simple consequence of this proof we obtain the following corollary.

Corollary 17 *Calculating the denotation of a chainless RSRL-formula in a finite RSRL-interpretation is PTIME-hard in the size of the interpretation.*

The denotation of an RSRL-formula is the set of those elements of the universe for which the formula holds true. Even if the formula is a sentence and it turns out that it is true for all elements, it can well be the case that subformulae have denotations that differ from the whole universe and the empty set. Therefore it is necessary to store intermediate results of subformulae. And this cannot be done with only logarithmic space available. It is therefore very likely that PTIME is the least upper bound. Thus it is basically the fact that RSRL is a description logic that leads to the complexity being a little worse compared to the result for classical logic.

The current section provides an answer to one of the two basic questions of descriptive complexity theory, namely the question about the complexity of the satisfaction relation. A natural question is to ask about the converse: Let K be a class of finite interpretations of chainless RSRL. If $K \in PTIME$, is K then definable by an RSRL-sentence? Standard techniques for classical logic answer this type of questions for *ordered* structures by simulating the computations of a Turing machine in a suitable logic. Trying to do the same for (chainless) RSRL is surprisingly difficult. At the moment it is even unclear, whether order can be axiomatised in RSRL. That is to say, suppose our signature contains a relation symbol $<$ — as is standardly assumed when simulating computations by a logic — it is simple to write down a few first order logic axioms that express that $<$ is indeed a linear order. Whether this can be done in RSRL, is an open question. The classical first order axioms can clearly not be used, and neither any simple variant of it. On the other hand, order is crucial in simulating computation steps — which are inherently ordered — in a logic. There are almost no results in descriptive complexity theory if the logic does not facilitate the axiomatisation of order. From such a perspective, it seems unlikely that Turing machine computations can be captured by RSRL. Hence there is no promising way to find a logical definition for a class of finite RSRL-structures that is defined by acceptability by a Turing machine of a certain type.

5 Conclusion

We presented in this paper a computability and a complexity result on RSRL. In a first part, we showed that truth of an RSRL-sentence in a finite RSRL-structure is in general undecidable by coding Post correspondence problems in RSRL. It follows that the denotation of a formula in a finite RSRL-structure is in general uncomputable. These results rely on the fact that relations range over and variables denote not only elements of the carrier but also finite lists of elements. This suggests a view of RSRL as a many-sorted logic with inherent polymorphism of relations and variables. The natural question to the linguists using RSRL is therefore whether this polymorphism is used at all or wether it is just an artefact of the way the semantics of RSRL is currently defined. In a many-sorted reformulation it becomes immediately visible that one of the denotation domains, namely the domain of finite lists, is infinite even if there are only finitely many different elements in the lists. The uncomputability result is quite normal under such a perspective, because it shows RSRL as a powerful language over lists, in which Post correspondence problems can easily be coded.

In a second part, we provided a complexity result for chainless RSRL. This is a weaker version of RSRL in which relations range over and variables denote only elements of the carrier, not lists. If K is a class of finite structures defined by a sentence of chainless RSRL then K is in PTIME, i.e., K is accepted by a deterministic Turing machine polynomially time bounded by the size of the structures. It follows that calculating the denotation of a chainless RSRL-formula in a finite structure is PTIME-hard in the size of the structure. These complexity results are not too far away from the ones for classical first order logic: Classes of first order structures definable by first order logic are in LOGSPACE. That chainless RSRL comes out with a higher complexity is attributed to the fact that (chainless) RSRL is a description logic. The denotation of a formula is not just true or false but rather the set of objects (elements of the carrier) for which the formula is true. When calculating the denotation of a complex formula it is necessary to handle intermediate results of subformulae which can in general not be stored on logarithmic space only.

Acknowledgements

I would like to thank Frank Morawietz, Uwe Mönnich, Frank Richter, and Manfred Sailer for helpful suggestions and interesting discussions.

References

[1] Alonso Church. A Note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1(40–41), 1936. Correction ibid. P. 101–102.

[2] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Springer-Verlag, 1995.

[3] Neil Immerman. Expressibility as a Complexity Measure: Results and Directions. In *Second Structure in Complexity Theory Conference*, pages 194–202. Computer Soc. of the IEEE, 1987.

[4] Stephan Kepser. A Satisfiability Algorithm for a Typed Feature Logic. Master's thesis, Seminar für Sprachwissenschaft, Universität Tübingen, Arbeitspapiere des SFB 340, Bericht Nr. 60, 1994.

[5] Paul John King. *A Logical Formalism for Head-Driven Phrase Struture Grammar*. PhD thesis, University of Manchester, 1989.

[6] Paul John King. Towards Truth in HPSG. In Valia Kordoni, editor, *Tübingen Studies in Head-Driven Phrase Structure Grammar, Vol 2*, pages 301–352. Arbeitspapiere des SFB 340, Bericht Nr. 132, 1999.

[7] Paul John King, Kiril Ivanov Simov, and Bjørn Aldag. The Complexity of Modellability in Finite and Computable Signatures of a Constraint Logic for Head-Driven Phrase Structure Grammar. *Journal of Logic, Language and Information*, 8(1):83–110, 1999.

[8] Carl Pollard and Ivan A. Sag. *Information Based Syntax and Semantics, Vol. 1: Fundamentals*. Number 13 in Lecture Notes. CSLI, 1987.

[9] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, 1994.

[10] Emil Post. A Variant of a Recursively Unsolvable Problem. *Bulletin of the AMS*, 52:264–268, 1946.

[11] Frank Richter. *A Mathematical Formalism for Linguistic Theories with an Application in Head-Driven Phrase Structure Grammar*. PhD thesis, SfS, Universität Tübingen, 2000.

[12] Frank Richter, Manfred Sailer, and Gerald Penn. A Formal Interpretation of Relations and Quantification in HPSG. In Gosse Bouma, Erhard Hinrichs, Geert-Jan M. Kruijff, and Richard T. Oehrle, editors, *Constraints and Resources in Natural Language Syntax and Semantics*, pages 281–298. CSLI Publications, 1999.