Characterising Space Complexity Classes via Knuth-Bendix Orders*

Guillaume Bonfante[1] and Georg Moser[2]

[1] INRIA-LORIA, Nancy, France, email: guillaume.bonfante@loria.fr
[2] Institute of Computer Science, University of Innsbruck, Austria, email: georg.moser@uibk.ac.at

Abstract. We study three different space complexity classes: LINSPACE, PSPACE, and ESPACE and give complete characterisations for these classes. We employ rewrite systems, whose termination can be shown by Knuth Bendix orders. To capture LINSPACE, we consider positively weighted Knuth Bendix orders. To capture PSPACE, we consider unary rewrite systems, compatible with a Knuth Bendix order, where we allow for padding of the input. And to capture ESPACE, we make use of a non-standard generalisation of the Knuth Bendix order.

1 Introduction

Term rewriting is a conceptually simple, but powerful abstract model of computation with applications in automated theorem proving, compiler optimisation, and declarative programming, to name a few. The *derivational complexity* of a term rewrite system (TRS for short) measures the complexity of a given TRS $\mathcal{R}$ by linking the maximal height of a given derivation over $\mathcal{R}$ to the size of the initial term. This notion was suggested by Hofbauer and Lautemann in [8]. Based on investigations of termination techniques, like simplification orders or the dependency pair method, a considerable number of results establish upper-bounds on the growth rate of the derivational complexity function. See for example [7, 18, 12, 14, 10, 16, 15] for a collection of results in this direction. We exemplary mention a result on the Knuth Bendix order (KBO for short). In [12] it is shown that KBO induced a tight 2-recursive upper bound on the derivational complexity. Note that this high upper bound on the complexity depends on the fact that we consider arbitrary TRSs. For unary TRS, Hofbauer has shown that the derivational complexity is bounded by an exponential function, cf. [6].

We are concerned here with a complementary point of view. The theme of our investigations is that of implicit characterisations of complexity classes: given a terminating TRS $\mathcal{R}$, can the information on the termination proof be applied to estimate the computational complexity of the functions computed by $\mathcal{R}$? For such investigations it is often possible to utilise results on the derivational complexity of a TRS. In particular in [4] the first author together with Cichon, Marion and Touzet established for example complete characterisations

* This research is partly supported by FWF (Austrian Science Fund) project P20133.

of the complexity classes PTIME and ETIME, together with their nondeterministic analogues. These results were obtained by a fine analysis of polynomially terminating TRS. In a similar vein [13, 5, 1, 2] establish the characterisations of various time and space complexity classes like LOGSPACE, PTIME, LINSPACE, and PSPACE. In this paper we focus on the space complexity classes LINSPACE, PSPACE, and ESPACE. In order to capture these classes we employ variants of KBO. More precisely we obtain the following results (all definitions will be given below):

- First, we completely capture LINSPACE through positively weighted KBOs on confluent TRS.
- Second, we completely capture PSPACE on unary TRS, compatible with KBO.
- Third, we completely capture ESPACE on unary TRS, compatible with enriched KBOs.

Moreover, we obtain similar characterisation of the nondeterministic analogues of these complexity classes, if the condition on confluence is dropped. Apart from the technical contribution these results are also of conceptional interest. On one hand they shows the versatility of KBO. On the other these results show how results on the derivational complexity of TRS (a measure on the *length* of derivations) are applicable to capture *space* complexity classes.

The paper is organised as follows. In Section 2 we recall basic notions form rewriting and the theory of computations. In Section 3 we define the notion of a relation computed by a rewrite systems, while in Section 4 we link this to the more standard notions of computation by a Turing machine. In Section 5, we recall the definition of KBO. Our main results are given in Section 6–8. Finally, we conclude in Section 9.

2 Preliminaries

We consider two ways of computing, the first one uses rewriting, the second one uses Turing machines. We assume familiarity with the basics of rewriting [3, 17], and with the basics of the theory of computation [11]. In this section we recall the central definitions for these areas.

Let $\mathcal{V}$ denote a countably infinite set of variables and $\mathcal{F}$ a signature. A signature is called *unary* if $\mathcal{F}$ consists of constants and unary function symbols only. The set of terms over $\mathcal{F}$ and $\mathcal{V}$ is denoted as $\mathcal{T}(\mathcal{F}, \mathcal{V})$. $\mathcal{V}\mathsf{ar}(t)$ denotes the set of variables occurring in a term t and $|t|_x$ denotes the number of occurrences of the variable x in t. A term t such that $\mathcal{V}\mathsf{ar}(t) = \varnothing$ is called *ground*. The *size* $|t|$ of a term is defined as the number of symbols in t.

A *term rewrite system* (*TRS* for short) $\mathcal{R}$ over $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is a *finite* set of rewrite rules $l \to r$, such that $l \notin \mathcal{V}$ and $\mathcal{V}\mathsf{ar}(l) \supseteq \mathcal{V}\mathsf{ar}(r)$. The induced rewrite relation is denoted by $\to_{\mathcal{R}}$. The transitive closure of $\to_{\mathcal{R}}$ is denoted by $\to^+_{\mathcal{R}}$, and its transitive and reflexive closure by $\to^*_{\mathcal{R}}$. A term $s \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ is called a *normal form* if there is no $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $s \to_{\mathcal{R}} t$. We write $s \to^!_{\mathcal{R}} t$ if

$s \to_{\mathcal{R}}^{*} t$ and t is in normal forms with respect to $\to_{\mathcal{R}}$. If no confusion can arise from this, we simply write $\to$ ($\to^*$, $\to^+$) instead of $\to_{\mathcal{R}}$ ($\to_{\mathcal{R}}^*$, $\to_{\mathcal{R}}^+$). We call a TRS *terminating* if no infinite rewrite sequence exists. Let s and t be terms. If exactly n steps are performed to rewrite s to t we write $s \to^n t$. A TRS is called *confluent* if for all $s, t_1, t_2 \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ with $s \to^* t_1$ and $s \to^* t_2$ there exists a term t_3 such that $t_1 \to^* t_3$ and $t_2 \to^* t_3$. A TRS $\mathcal{R}$ is said to be *unary* if $\mathcal{R}$ is based on a unary signature. The *derivation height* of a terminating term s (with respect to $\mathcal{R}$) is defined as $\mathsf{dh}(s) := \max\{n \mid \exists\, t \text{ such that } s \to_{\mathcal{R}}^{n} t\}$. The *derivational complexity function* (with respect to $\mathcal{R}$) is defined as follows: $\mathsf{dc}_{\mathcal{R}}(n) = \max\{\mathsf{dh}(t) \mid |t| \leqslant n\}$. A *proper order* is a transitive and irreflexive relation and a *preorder* is a transitive and reflexive relation. A proper order $\succ$ is *well-founded* if there is no infinite decreasing sequence $t_1 \succ t_2 \succ t_3 \cdots$.

Notation. Let $\mathcal{F}$ be a unary signature. In the sequel of the paper, we sometimes confuse the notation of elements of $\mathcal{T}(\mathcal{F}, \mathcal{V})$ as terms or strings. For example, for $\mathcal{F} = \{\mathsf{f}, \mathsf{g}, \bullet\}$, where f, g are unary and $\bullet$ is a constant, we may denote the ground term $\mathsf{f}(\mathsf{g}(\mathsf{f}(\bullet)))$ as the string fgf. We may even mix both representations, for example, we may write $\mathsf{f}(\mathsf{gg})$, instead of $\mathsf{f}(\mathsf{g}(\mathsf{g}(\bullet)))$. No confusion will arise from this.

We fix the representation of *Turing machines* (*TMs* for short). A *deterministic one-tape Turing machine* is a 9-tuple $\mathsf{M} = (Q, \Sigma, \Gamma, \triangleright, \sqcup, \delta, \mathrm{s}, \mathrm{t}, \mathrm{r})$ such that Q is a finite set of *states*, Σ is a finite *input alphabet*, Γ is a finite *tape alphabet*, $\Sigma \subseteq \Gamma$, $\sqcup \in \Gamma$ is the *blank symbol*, $\triangleright \in \Gamma$ is the *left endmarker*, $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the *transition function*, and $\mathrm{s}, \mathrm{t}, \mathrm{r} \in Q$ are the *start state*, the *accept state*, and the *reject state*, respectively. (As usual, we demand $\mathrm{t} \neq \mathrm{r}$.) We assert that the left endmarker $\triangleright$ is never overwritten and the machine never moves off the left tape. Moreover, we require that once the machine enters an accept (reject) state, it must neither leave this state nor modify the tape. In order to define *nondeterministic* TMs, we replace the transition function δ by a relation Δ. We say a TM *halts* if it either reaches the accepting state t, or the rejecting state r.

A configuration of a TM M is given by a triple $(q, v\sqcup^\infty, n)$ with $q \in Q$ being the current state of the machine, $v \in \Gamma^*$ being the current content of the tape and n denotes the position of the head on the tape. (Here $\sqcup^\infty$ denotes an infinite number of blanks.) The *start configuration* on input $w \in \Sigma^*$ is defined as $(\mathrm{s}, \triangleright w\sqcup^\infty, 1)$. The *next configuration relation* of M is denoted as $\xrightarrow{\mathsf{M}}$.

Let $\xrightarrow{\mathsf{M}^*}$ denote the reflexive and transitive closure of the next configuration relation $\xrightarrow{\mathsf{M}}$. We say M *accepts* its input w if $(\mathrm{s}, \triangleright w\sqcup^\infty, 1) \xrightarrow{\mathsf{M}^*} (\mathrm{t}, v, n)$ for some $v \in \Gamma^*$, $n \in \mathbb{N}$. The language of M, that is the set of words it accepts, is denoted as $\mathsf{L}(\mathsf{M})$. We say M *decides* a language L if $L = \mathsf{L}(\mathsf{M})$ and M halts on all inputs. Furthermore M is said to *decide a binary relation* R if M decides the language $L = \{(w, v) \mid (w, v) \in R\}$. Finally, let R be a binary relation such that $R(w, v)$ can be decided by a (nondeterministic) TM. Then the *function problem* associated with R (denoted as F_R) is the problem to find a string v such that

$R(w, v)$ holds, given the string w. Or reject, if no such v exists. For a complexity class $\mathcal{C}$, we write $\mathsf{F}\mathcal{C}$ to denote the function problems associated with this class.

We recall the definition of the complexity classes considered in the sequel. We say that a TM *runs in space* $S(n)$ or is $S(n)$ space-bounded, if on all but finitely many inputs w, no computation path uses more than $S(|w|)$ tape cells.

For any bounding function $S\colon \mathbb{N} \to \mathbb{N}$, such that $S(n) = \Omega(n)$, we define: $\mathsf{DSPACE}(S(n)) := \{\mathsf{L}(\mathsf{M}) \mid \mathsf{M} \text{ is a } S(n) \text{ space-bounded deterministic TM}\}$. The following definitions are well-known: $\mathsf{LINSPACE} := \bigcup_{k>0} \mathsf{DSPACE}(k \cdot n)$, $\mathsf{PSPACE} := \bigcup_{k>0} \mathsf{DSPACE}(n^k)$, $\mathsf{ESPACE} := \bigcup_{k>0} \mathsf{DSPACE}(2^{k \cdot n})$. $\mathsf{NLINSPACE}$, $\mathsf{NPSPACE}$, and $\mathsf{NESPACE}$, respectively, denote the corresponding nondeterministic complexity classes. Recall the following equalities: $\mathsf{PSPACE} = \mathsf{NPSPACE}$, $\mathsf{ESPACE} = \mathsf{NESPACE}$, see [11].

3 Computation by Rewriting

In this section we introduce what is meant by the relation (or function) computed by a (confluent) TRS. We essentially follow the approach in [4].

Let Σ be a fixed finite alphabet and let ϵ denote the empty string. We define the signature $\mathcal{F}(\Sigma)$ *corresponding* to Σ as follows: for all $a \in \Sigma$ let $\alpha(a) \in \mathcal{F}(\Sigma)$ denote a unary function symbol. In addition $\mathcal{F}(\Sigma)$ contains a unique constant symbol $\bullet$. The mapping α is lifted to a function α between strings Σ^* and $\mathcal{T}(\mathcal{F}, \mathcal{V})$ as follows: (i) $\alpha(\epsilon) := \bullet$, (ii) $\alpha(av) := \alpha(a)(\alpha(v))$. The mapping α is called *translation from* Σ *to* $\mathcal{F}(\Sigma)$. The translation between $\mathcal{F}(\Sigma)$ and Σ, that is, the reversal of the mapping α, is defined analogously.

Definition 1. *Let Σ be an alphabet and let $\mathcal{F}(\Sigma)$ denote the signature corresponding to Σ. Suppose $\mathcal{R}$ is a terminating TRS over signature $\mathcal{F} \supseteq \mathcal{F}(\Sigma)$. Let $\mathsf{No} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V})$ be a finite set of* non-accepting *terms. A relation $R \subseteq \Sigma^* \times \Sigma^*$ is* computable by $\mathcal{R}$ *if there exists a function symbol $\mathsf{f} \in \mathcal{F}$ such that for all $w \in \Sigma^*$: $R(w, \mathtt{out}(t))$ if and only if $\mathsf{f}(\mathtt{inp}(w)) \to_{\mathcal{R}}^{!} t$ such that $t \notin \mathsf{No}$. Here $\mathtt{inp}$, $\mathtt{out}$ denote translation from Σ and into Σ, respectively. To sum up, we say that R is computed by the 7-tuple $(\Sigma, \mathcal{F}, \mathtt{inp}, \mathtt{out}, \mathsf{f}, \mathcal{R}, \mathsf{No})$ or shorter by $\mathcal{R}$ when the other components are clear from the context.*

Suppose $\mathcal{R}$ is *confluent* and computes a relation R. For this case, we also say that $\mathcal{R}$ computes the (partial) *function* induced by the relation R. Note that the requirement $t \notin \mathsf{No}$ is a technical one and serves only to characterise an *accepting run* of a TRS $\mathcal{R}$. A typical example of its use would be to set $\mathsf{No} = \{\mathsf{undef}\}$ if the TRS contains a rule of the form $l \to \mathsf{undef}$ and undef is not part of the relation to be defined. If No is clear from context, we will not speak about it.

In the sequel of the section, we provide several examples of (unary) TRS together with the binary relation computed. Given a word w on some alphabet $\Sigma \cup \{b\}$, $\mathsf{rem}(b, w)$ denotes the word w where the letter b has been removed. We define: (i) $\mathsf{rem}(b, \epsilon) := \epsilon$, (ii) $\mathsf{rem}(b, bw) := \mathsf{rem}(b, w)$, (iii) $\mathsf{rem}(b, aw) := a\ \mathsf{rem}(b, w)$, if $a \neq b$. By extension, $\mathsf{rem}(\{b_1, \ldots, b_k\}, w) = \mathsf{rem}(b_1, \mathsf{rem}(b_2, \ldots, \mathsf{rem}(b_k, w) \cdots))$. Given a finite alphabet Σ and an extra letter $b \notin \Sigma$, the TRS $\mathcal{R}_{\leftarrow \mathsf{b}}$ defined in

the next example, re-orders the symbols of a word by shifting the letter b on the left of the word. More precisely, $(\Sigma \cup \{b\}, \mathcal{F}(\Sigma \cup \{b\}), \mathtt{inp}, \mathtt{out}, \mathsf{f}_{\leftarrow \mathsf{b}}, \mathcal{R}_{\leftarrow \mathsf{b}}, \mathsf{No})$ computes the relation $S \subseteq (\Sigma \cup \{b\})^* \times (\Sigma \cup \{b\})^*$ containing all $(w, w') \in (\Sigma \cup \{b\})^* \times (\Sigma \cup \{b\})^*$ such that $w' = b^k \mathsf{rem}(b, w)$ with $k \geqslant 0$ and $|w| = |w'|$.

Example 2. Let $\mathtt{inp}(b) = \mathsf{b}$ and let $\mathcal{F} = \mathcal{F}(\Sigma) \cup \{\mathsf{b}, [\![_{\mathsf{b}},]\!]_{\mathsf{b}}, \mathsf{f}, \mathsf{f}_{\leftarrow \mathsf{b}}\}$, where all function symbols but $\bullet \in \mathcal{F}(\Sigma)$ are unary. Consider the TRS $\mathcal{R}_{\leftarrow \mathsf{b}}$ defined as follows:

$$\begin{array}{lll} \mathsf{f}_{\leftarrow \mathsf{b}}(x) \to [\![_{\mathsf{b}}(\mathsf{f}(x)) & g(\mathsf{b}(x)) \to \mathsf{b}(g(x)) & [\![_{\mathsf{b}}(\mathsf{b}(x)) \to \mathsf{b}([\![_{\mathsf{b}}(x)) \\ \mathsf{f}(h(x)) \to h(\mathsf{f}(x)) & g(]\!]_{\mathsf{b}}(x)) \to]\!]_{\mathsf{b}}(g(x)) & [\![_{\mathsf{b}}(]\!]_{\mathsf{b}}(x)) \to x \\ \mathsf{f}(\bullet) \to]\!]_{\mathsf{b}}(\bullet)\ , & & \end{array}$$

where $g \in \mathcal{F}(\Sigma)$, $h \in \mathcal{F}(\Sigma \cup \{b\})$. Let $\mathtt{inp} = \mathtt{out} : d \in \Sigma \cup \{b\} \mapsto d$ and $\mathsf{No} = \varnothing$. It is not difficult to see that the TRS $\mathcal{R}_{\leftarrow \mathsf{b}}$ is confluent.

Some comments about this system: $[\![_{\mathsf{b}},]\!]_{\mathsf{b}}$ serve respectively as the left and the right end markers of the symbol b. Along the computation, any occurrence of the symbol b is at the left of $]\!]_{\mathsf{b}}$ and any symbol $g \in \mathcal{F}(\Sigma)$ is at the right of $[\![_{\mathsf{b}}$. At any step of a derivation $\mathsf{f}_{\leftarrow \mathsf{b}}(\mathtt{inp}(w)) \to^+ t$, we have: (i) $\mathsf{rem}(\{[\![_{\mathsf{b}},]\!]_{\mathsf{b}}, \mathsf{f}\}, t) = w$, and (ii) either $t = b^k([\![_{\mathsf{b}}(v))$ for some $k \geqslant 0$ and some term v, or t does not contain $[\![_{\mathsf{b}}$ and t is the normal form with respect to $\mathcal{R}_{\leftarrow \mathsf{b}}$. That is, $t = b^k \mathsf{rem}(b, w)$ with k suitable such that $|t| = |w|$. Let $\#(b, w)$ denote the number of occurrence of b in w. Due to these two invariants, in the second clause, when $t = b^k([\![_{\mathsf{b}}(v))$, we can state that $k \leqslant \#(b, w)$. In other words, all along the computation, the prefix in front of $[\![_{\mathsf{b}}$ is a prefix of the normal form. Similar invariants and considerations can be applied to show that the TRS $\mathcal{R}_{\to \mathsf{b}}$, defined in the next example, operates as claimed.

Example 3. Let $\mathcal{F} = \mathcal{F}(\Sigma) \cup \{\mathsf{b}, [\![_{\Sigma},]\!]_{\Sigma}, \mathsf{f}, \mathsf{f}_{\to \mathsf{b}}\}$. In an analogous way, the TRS $\mathcal{R}_{\to \mathsf{b}}$ shifts the symbols b to the right.

$$\begin{array}{lll} \mathsf{f}_{\to \mathsf{b}}(x) \to [\![_{\Sigma}(\mathsf{f}(x)) & \mathsf{b}(g(x)) \to g(\mathsf{b}(x)) & [\![_{\Sigma}(g(x)) \to g([\![_{\Sigma}(x)) \\ \mathsf{f}(h(x)) \to h(\mathsf{f}(x)) & \mathsf{b}(]\!]_{\Sigma}(x)) \to]\!]_{\Sigma}(\mathsf{b}(x)) & [\![_{\Sigma}(]\!]_{\Sigma}(x)) \to x \\ \mathsf{f}(\bullet) \to]\!]_{\Sigma}(\bullet)\ , & & \end{array}$$

where $g \in \mathcal{F}(\Sigma)$, $h \in \mathcal{F}(\Sigma \cup \{b\})$.

We end the section with a more liberal notion of computability by rewriting, computations up-to a *padding* function.

Definition 4. *Let $\Sigma, \mathcal{F}(\Sigma), \mathcal{R}$ and No be as in Definition 1. Furthermore, we suppose the signature $\mathcal{F} \supset \mathcal{F}(\Sigma)$ contains a fresh padding symbol, denoted by $\sqcap$. Let $F : \mathbb{N} \to \mathbb{N}$ be a function. A relation $R \subseteq \Sigma^* \times \Sigma^*$ is* computable by $\mathcal{R}$ *up-to the* padding function F*, if there exists a function symbol $\mathsf{f} \in \mathcal{F}$ such that for all $w \in \Sigma^*$: $R(w, \mathtt{out}(t))$ if and only if $\mathsf{f}(\mathtt{inp}(w) \sqcap^{F(|x|)}) \to^!_{\mathcal{R}} t$, such that $t \notin \mathsf{No}$. Here $\mathtt{inp}$, $\mathtt{out}$ denote translations from Σ and into Σ as above.*

We say that a relation R is computable up-to F-padding, if it is computable by some $\mathcal{R}$ up-to the padding function F.

Example 5. Consider an alphabet Σ of unary symbols and a symbol $b \notin \Sigma$. On $(\Sigma \cup \{b\})^*$, we define a relation R as follows: $R(w, v)$ if and only if $v = b^{|w|}\ w$, such that w does not contain b. Consider the TRS $\mathcal{R}$, obtained by adding the following rules to the TRS $\mathcal{R}_{\leftarrow \mathsf{b}}$:

$$\begin{array}{rrr} \mathsf{g}'(x) \to \mathsf{f}_{\leftarrow \mathsf{b}}(\mathsf{ver}(x)) & \mathsf{ver}(g(x)) \to g(\mathsf{ver}(x)) & h(\mathsf{undef}) \to \mathsf{undef} \\ \mathsf{ver}(\sqcap(x)) \to \mathsf{b}(\mathsf{ver}(x)) & \mathsf{ver}(\mathsf{b}(x)) \to \mathsf{undef} & \mathsf{ver}(\bullet) \to \bullet\,, \end{array}$$

where $g \in \mathcal{F}(\Sigma), h \in \mathcal{F} \setminus \{\bullet\}$. Then R is computable by $\mathcal{R}$ by padding up-to the polynomial $\lambda x.x$.

4 Turing Machines and Rewriting

In this section, we introduce a simulation of Turing machines by rewriting.

Let $\mathsf{M} = (Q, \Sigma, \Gamma, \triangleright, \sqcup, \Delta, \mathrm{s}, \mathrm{t}, \mathrm{r})$ denote a (nondeterminstic) TM. We define the signature $\mathcal{F}(\mathsf{M})$ *corresponding* to M as follows. For each $a \in \Gamma$, there exists a unary function symbol $\mathsf{a} \in \mathcal{F}(\mathsf{M})$. Moreover define $Q(\mathcal{F}) := \{\mathsf{q}_\mathsf{a} \mid q \in Q \text{ and } a \in \Gamma\}$ and suppose $Q(\mathcal{F}) \subseteq \mathcal{F}(\mathsf{M})$. Finally we have $\bullet, \mathsf{0}, \mathsf{1} \in \mathcal{F}(\mathsf{M})$.

It is well-known that TMs can be encoded as unary TRSs or even string rewrite systems, cf. [17]. In the following encoding, we make use of the fact that all here considered TMs are space bounded by some bounding function S. Thus we need only consider a finite part of the tape of M. In particular any configuration α of M becomes representable as a triple $(q, \triangleright v \sqcup^m, \ell)$ for $m \in \mathbb{N}$. This is employed in the definition given belows. To represent the running time of M, we add a time point to the representation of the configuration. It is well-known that a halting machine working in space $S(n)$ runs in time $2^{k \cdot S(n)}$ for some $k > 0$ (see for example [11]). Thus, the running time can be represented in binary by a word of length $k \cdot S(n)$. Let w be a word of length $n = |w|$; we write w_i $(1 \leqslant i \leqslant n)$ to denote the i^{th} letter in w.

Definition 6. *Let w be a word over Σ and let $n = |w|$. Suppose M is a (nondeterministic) TM running on input w in space $S(n)$ such that $S(n) = \Omega(n)$. Further let $\alpha = (q, \triangleright v \sqcup^m, \ell)$ be a configuration of M at time t such that $|v| + m \leqslant S(n)$. Then α is represented as: $\gamma_{t,q,v,\ell} := t \triangleright v_1 v_2 \dots v_{\ell-1} q_{v_\ell} v_{\ell+1} v_{\ell+2} \dots v_{|v|} \sqcup^m$. The time point t is represented as a string of length $k \cdot S(n)$, making use of the extra symbols $\mathsf{0}, \mathsf{1} \in \mathcal{F}(\mathsf{M})$.*

Lemma 7 (Step by step simulation of a TM). *Let S be a bounding function such that $S(n) = \Omega(n)$. Suppose we are given a nondeterminstic TM $\mathsf{M} = (Q, \Sigma, \Gamma, \triangleright, \sqcup, \Delta, s, t, r)$ running with input w in space $S(|w|)$. Then there exists a unary TRS $\mathcal{R}$, simulating this machine step by step, starting with $\gamma_{t_0,s,w,1}$, where $t_0 = \mathsf{0}^{S(|w|)}$. More precisely: if $(p, w, m) \xrightarrow{\mathsf{M}} (q, v, n)$, then $\gamma_{t,p,w,m} \to^*_{\mathcal{R}} \gamma_{t+1,q,v,n}$ for $t \geqslant 0$.*

Proof. We add the extra symbol $\mathsf{0}'$ to the signature to manage the clock (see the rules below). Let $\mathcal{F}$ be a signature that extends the signature $\mathcal{F}(\mathsf{M})$ so that $\{\mathsf{a}' \mid a \in \Gamma\} \cup \{\mathsf{0}'\} \subseteq \mathcal{F}$.

Consider the following (schematic) TRS $\mathcal{R}_1$ over $\mathcal{F}$. Essentially $\mathcal{R}_1$ computes the binary successor and in addition marks letters. In conjunction with rules that express each transition of M it will be used to update the configuration.

$$\begin{aligned} \mathsf{0}(f(x)) &\to \mathsf{1}(f'(x)) \qquad \mathsf{1}(f(x)) \to \mathsf{0}'(f'(x)) \qquad f'(g(x)) \to f(g'(x)) \\ \mathsf{0}(\mathsf{0}'(x)) &\to \mathsf{1}(\mathsf{0}(x)) \qquad \mathsf{1}(\mathsf{0}'(x)) \to \mathsf{0}'(\mathsf{0}(x)) \;, \end{aligned}$$

where $f, g \in \mathcal{F}$ and f', g' denote marked copies of f, g, respectively. The next (schematic) rules represent the transition relation Δ of M.

$$\begin{aligned} \mathsf{a}'(\mathsf{p}_\mathsf{b}(x)) &\to \mathsf{q}_\mathsf{a}(\mathsf{c}(x)) && \text{if } (q, c, L) \in \Delta(p, b) \\ \mathsf{a}'(\mathsf{p}_\mathsf{b}(\mathsf{d}(x))) &\to \mathsf{a}(\mathsf{c}(\mathsf{q}_\mathsf{d}(x))) && \text{if } (q, c, R) \in \Delta(p, b) \end{aligned}$$

Here $\mathsf{a}, \mathsf{a}', \mathsf{c}, \mathsf{d} \in \mathcal{F}$. These rules are collected in the TRS $\mathcal{R}_2$. Finally, we define $\mathcal{R} := \mathcal{R}_1 \cup \mathcal{R}_2$. It is not difficult to see that $\mathcal{R}$ fulfils the conditions given in the lemma. Note that this simulation takes square time. More precisely, if M works in $T(n)$ (nondeterministic) steps, the derivation length of the rewriting system will be $\mathrm{O}(T(n)^2)$. □

5 The Knuth Bendix Order

In this section we recall the definition of the Knuth Bendix order. We follow the presentation in [3] closely.

Let $\succ$ be a proper order on signature $\mathcal{F}$, called *precedence*. A *weight function* for $\mathcal{F}$ is a pair (w, w_0) consisting of a function $\mathsf{w}\colon \mathcal{F} \to \mathbb{N}$ and a minimal weight $w_0 \in \mathbb{N}$, $w_0 > 0$ such that $\mathsf{w}(c) \geqslant w_0$ if c is a constant. A weight function (w, w_0) is called *admissible* for a precedence $\succ$ if the existence of $f \in \mathcal{F}$, such that f is unary and $\mathsf{w}(f) = 0$ entails that $f \succ g$ for all $g \in \mathcal{F}$, $g \neq f$. The symbol f is necessarily unique, it is qualified as *special*. We sometimes identify the pair (w, w_0) with the function $\mathsf{w}\colon \mathcal{F} \to \mathbb{N}$ and simply call the latter a weight function. The *weight* of a term t, denoted as $\mathsf{weight}(t)$ is defined inductively. Assume t is a variable, then set $\mathsf{weight}(t) := w_0$, otherwise if $t = f(t_1, \dots, t_n)$, we define $\mathsf{weight}(t) := \mathsf{w}(f) + \mathsf{weight}(t_1) + \cdots + \mathsf{weight}(t_n)$.

Definition 8. *Let $\succ$ denote a precedence and (w, w_0) an admissible weight function. The* Knuth Bendix order *(*KBO *for short) $\succ_{\mathsf{kbo}}$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is inductively defined as follows: for all terms s, t we have $s \succ_{\mathsf{kbo}} t$ if $|s|_x \geqslant |t|_x$ for all $x \in \mathcal{V}$ and either*

1) $\mathsf{weight}(s) > \mathsf{weight}(t)$, *or*
2) $\mathsf{weight}(s) = \mathsf{weight}(t)$, *and one of the following alternatives holds:*
   (a) *t is a variable, $s = f^n(t)$, where f is the special symbol, and $n > 0$,*
   (b) *$s = f(s_1, \dots, s_n)$, $t = f(t_1, \dots, t_n)$, and there exists $i \in \{i, \dots, n\}$ such that $s_j = t_j$ for all $1 \leqslant j < i$ and $s_i \succ_{\mathsf{kbo}} t_i$, or*

(c) $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_m)$, *and* $f \succ g$.

Let $\succcurlyeq_{\mathsf{kbo}}$ *denote the quasi-order induced by* $\succ_{\mathsf{kbo}}$.

We say a TRS $\mathcal{R}$ is *compatible* with a KBO $\succ_{\mathsf{kbo}}$ if $\mathcal{R} \subseteq \succ_{\mathsf{kbo}}$, that is for all rules $l \to r$: $l \succ_{\mathsf{kbo}} r$. It is well-known that KBO is a simplification order, in particular if $\mathcal{R}$ is compatible with $\succ_{\mathsf{kbo}}$, then $\mathcal{R}$ terminates. A KBO induced by a weight function $\mathsf{w}\colon \mathcal{F} \to \mathbb{N} \setminus \{0\}$ is said to be *positively weighted*.

Example 9. Recall Example 2. Let $\mathsf{w}(f) = 1$ for all symbols in the TRS $\mathcal{R}_{\leftarrow\mathsf{b}}$, with the exception of $\mathsf{f}_{\leftarrow\mathsf{b}}$, where we set $\mathsf{w}(\mathsf{f}_{\leftarrow\mathsf{b}}) = 3$. Consider a precedence $\succ$ defined as follows: $\mathsf{f} \succ [\![_{\mathsf{b}} \succ \mathcal{F}(\Sigma) \succ \mathsf{b} \succ]\!]_{\mathsf{b}} \succ \bullet$, where $[\![_{\mathsf{b}} \succ \mathcal{F}(\Sigma) \succ \mathsf{b}$ abbreviates that for any function symbol $g \in \mathcal{F}(\Sigma)$, $[\![_{\mathsf{b}} \succ g \succ \mathsf{b}$, and $\succ$ is defined arbitrary on $\mathcal{F}(\Sigma)$. It is easy to see that the KBO induced by the (admissible) weight function w and $\succ$ is compatible with $\mathcal{R}_{\leftarrow\mathsf{b}}$. Now, consider the TRS $\mathcal{R}_{\to\mathsf{b}}$ defined in Example 3. Let $\mathsf{w}(f) := 1$ for all symbols in the TRS $\mathcal{R}_{\leftarrow\mathsf{b}}$, with the exception of $\mathsf{f}_{\to\mathsf{b}}$, where we set $\mathsf{w}(\mathsf{f}_{\to\mathsf{b}}) := 3$. Consider a precedence $\succ$ defined as follows: we assert $\mathsf{f} \succ [\![_{\Sigma} \succ \mathsf{b}$, together with $\mathsf{b} \succ \mathcal{F}(\Sigma) \succ]\!]_{\Sigma} \succ \bullet$. Then $\mathcal{R}_{\to\mathsf{b}} \subseteq \succ_{\mathsf{kbo}}$, for the induced KBO $\succ_{\mathsf{kbo}}$.

The next example clarifies that the simulating TRS $\mathcal{R}$ defined in the proof of Lemma 7 is KBO terminating.

Example 10. Consider the TRS $\mathcal{R}$ over $\mathcal{F}$ defined in the proof of Lemma 7. Let $\mathsf{w}(f) := 1$ for all $f \in \mathcal{F}$. For the precedence $\succ$ we assert $\mathsf{0} \succ \mathsf{1} \succ \mathsf{0}'$ and for all $a, a' \in \mathcal{F}$ and all $q \in Q$ set $\mathsf{a}' \succ \mathsf{a}$, $\mathsf{a}' \succ \mathsf{q}_{\mathsf{b}}$, where $b \in \Sigma$. It is easy to see that $\mathcal{R}$ is compatible with the KBO induced by the given weight function and the precedence.

In the next sections, we discuss a variety of variants of KBO in order to classify their computational content precisely. For that we introduce the following technical definition. The *σ-size* of a term t measures the number of non-special non-constant function symbols occurring in t: (i) $|t|_\sigma := 0$, if t is a constant; (ii) $|t|_\sigma := |t'|_\sigma$, if $t = f(t')$, f special; (iii) $|t|_\sigma := 1 + \sum_1^n |t_i|_\sigma$, if $t = f(t_1, \ldots, t_n)$.

Lemma 11. *Let* w *denote a weight function and* weight *the induced weight function on terms. For all terms* t*, the following hold: (i)* $|t|_\sigma = \mathrm{O}(\mathsf{weight}(t))$ *and (ii)* $\mathsf{weight}(t) = \mathrm{O}(|t|)$.

In concluding this section, we show how a function f computed by a TM running in space $S(n)$, where S is at least linear, is computed (up to S) by a TRS $\mathcal{R}$, whose termination can be shown via KBO.

Suppose $S(n) = k \cdot n + l$ and let R be computed by the 7-tuple $(\Sigma, \mathcal{F} \cup \{\sqcap\}, \mathsf{inp}, \mathsf{out}, \mathsf{g}, \mathcal{R}, \mathsf{No})$ up-to the padding function $S(n)$. In the following we define the 7-tuple $(\Sigma, \mathcal{F}', \mathsf{inp}', \mathsf{out}, \mathsf{g}', \mathcal{R}', \mathsf{No})$.

Define $\mathcal{R}_{\to\sqcap}$ as in Example 3, so that the symbol b is specialised to the padding symbol $\sqcap \in \mathcal{F}$. Set $\mathcal{F}_0 = \{\mathsf{f}_{\to\sqcap}, \mathsf{f}, [\![_{\Sigma},]\!]_{\Sigma}\}$. Without loss of generality, we assert that $\mathcal{F}_0$ and $\mathcal{F}$ are disjoint. Let $\mathcal{F}_1 = \{\mathsf{g}', \mathsf{cpy}_3\}$ be disjoint with $\mathcal{F}$ and let

$\overline{\Sigma}$ denote a copy of the alphabet Σ. Consider the auxiliary TRS $\mathcal{R}_{\mathsf{cpy}}$ over the signature $\mathcal{F}_1 \cup \mathcal{F}(\Sigma) \cup \mathcal{F}(\overline{\Sigma})$.

$$\mathsf{g}'(x) \to \mathsf{g}(\mathsf{f}_{\to\sqcap}(\mathsf{cpy}_3(x))) \quad \mathsf{cpy}_3(\bullet) \to \sqcap^l(\bullet) \quad \mathsf{cpy}_3(\overline{f}(x)) \to f(\sqcap^k(\mathsf{cpy}_3(x))) \;,$$

where $f \in \mathcal{F}(\Sigma)$ and $\overline{f} \in \mathcal{F}(\overline{\Sigma})$ its copy. Define $\mathcal{F}' = (\mathcal{F} \setminus \{\sqcap\}) \cup \mathcal{F}_0 \cup \mathcal{F}_1 \cup \mathcal{F}(\overline{\Sigma})$ and let $\mathcal{R}'$ be defined over $\mathcal{F}'$ as follows: $\mathcal{R}' := \mathcal{R} \cup \mathcal{R}_{\to\sqcap} \cup \mathcal{R}_{\mathsf{cpy}}$. The translation $\mathtt{inp}'$ from Σ is defined as follows:

$$\mathtt{inp}' \colon a \in \Sigma \mapsto \overline{a} \in \mathcal{F}(\overline{\Sigma}) \;.$$

This completes the definition of the 7-tuple $(\Sigma, \mathcal{F}', \mathtt{inp}', \mathtt{out}, \mathsf{g}', \mathcal{R}', \mathsf{No})$.

Lemma 12. *Let R be computed by the 7-tuple $(\Sigma, \mathcal{F} \cup \{\sqcap\}, \mathtt{inp}, \mathtt{out}, \mathsf{g}, \mathcal{R}, \mathsf{No})$ up-to the padding function $\lambda x.k{\cdot}x{+}l$, using the padding symbol $\sqcap$. Then, R is also computed by the 7-tuple $(\Sigma, \mathcal{F}', \mathtt{inp}', \mathtt{out}, \mathsf{g}', \mathcal{R}', \mathsf{No})$ defined above. Moreover, if $\mathcal{R}$ is compatible with a positively weighted KBO, the same holds with respect to $\mathcal{R}'$.*

Theorem 13. *Let $R(w, v)$ be a binary relation, let $n = |w|$ and let M be a (nondeterministic) TM that solves the function problem associated with relation $R(w, v)$ in space $S(n) = \Omega(n)$ and time $2^{S(n)}$. Then the relation R is computable by $\mathcal{R}$ up-to S-padding using the padding symbol $\sqcap$. Moreover $\mathcal{R}$ is unary and compatible with a positively weighted KBO.*

Proof. In proof, we assume M is nondeterminstic and we utilise the step by step simulation of a nondeterminstic TM $\mathsf{M} = (Q, \Sigma, \Gamma, \triangleright, \sqcup, \Delta, \mathrm{s}, \mathrm{t}, \mathrm{r})$ as seen in Lemma 7. The main issue is to properly prepare the "data", that is, the initial configuration, so that one can apply the lemma. Let the TRS $\mathcal{R}_{\mathsf{M}}$ over the signature $\mathcal{F}$ be defined as in the proof of Lemma 7. Furthermore define $\mathcal{R}_{\leftarrow 0}$ as in Example 2, such that the symbol b is specialised to the clock symbol $\mathsf{0} \in \mathcal{F}$. Moreover consider the following auxiliary TRS $\mathcal{R}_{\mathsf{build}}$:

$$\begin{aligned}
\mathsf{build}(x) &\to \triangleright'(\mathsf{f}_{\leftarrow 0}(\mathsf{cpy}_1(x))) & \mathsf{cpy}_1(f(x)) &\to f(\mathsf{cpy}_1(x)) \\
\mathsf{cpy}_1(\sqcap(x)) &\to \mathsf{0}(\sqcap(\mathsf{cpy}_1(x))) & \mathsf{cpy}_1(\bullet) &\to \bullet \\
\triangleright'(\mathsf{0}(x)) &\to \mathsf{0}(\triangleright'(x)) & \triangleright'(g(x)) &\to \triangleright(\mathrm{s}_{\mathsf{g}}(x)) \;,
\end{aligned}$$

where $f, g \in \mathcal{F}(\Sigma)$. Finally, we set $\mathcal{R} := \mathcal{R}_{\mathsf{M}} \cup \mathcal{R}_{\leftarrow 0} \cup \mathcal{R}_{\mathsf{build}}$. It is not difficult to see that $\mathcal{R}$ computes the relation R up-to S-padding.

We extend the definitions of weight and precedence as given in Example 10 for $\mathcal{R}_{\mathsf{M}}$ and in Example 9 for $\mathcal{R}_{\leftarrow 0}$. Let $\mathcal{F}_{\mathsf{M}}$ denote the signature of $\mathcal{R}_{\mathsf{M}}$ and let $\mathsf{w}(f) := 1$ for all $f \in \mathcal{F}_{\mathsf{M}}$. Moreover, let $\mathcal{F}_0 = \{[\![_0,]\!]_0, \mathsf{f}, \mathsf{f}_{\leftarrow 0}\}$ and note that the signature of $\mathcal{R}_{\leftarrow 0}$ amounts to $\mathcal{F}(\Sigma) \cup \mathcal{F}_0 \cup \{\mathsf{0}\}$. Observe that $\mathcal{F}(\Sigma) \cup \{\mathsf{0}\} \subseteq \mathcal{F}_{\mathsf{M}}$, that is, those weights have already been assigned. Set $\mathsf{w}([\![_0) = \mathsf{w}(]\!]_0) = \mathsf{w}(\mathsf{f}) := 1$ and $\mathsf{w}(\mathsf{f}_{\leftarrow 0}) = 2$. Finally consider the new symbols in (the signature) of $\mathcal{R}_{\mathsf{build}}$ and set $\mathsf{w}(\sqcap) := 2$, $\mathsf{w}(\triangleright') = \mathsf{w}(\mathsf{cpy}_1) = 1$, and $\mathsf{w}(\mathsf{build}) := 4$. This completes the definition of w for the signature of $\mathcal{R}$. Moreover extend the precedence $\succ$ by $\mathsf{build} \succ \triangleright' \succ \mathsf{f}_{\leftarrow 0} \succ \mathsf{cpy}_1 \succ \mathcal{F}(\Sigma)$ and $\mathsf{cpy}_1 \succ \mathsf{0}$. It is not difficult to see that the KBO $\succ_{\mathsf{kbo}}$ thus induced is compatible with KBO; furthermore $\succ_{\mathsf{kbo}}$ is positively weighted. ◻

6 Characterising Linear Space via KBO

In this section, we show how to capture the complexity classes LINSPACE and its nondeterministic variant NLINSPACE with the help of a restriction of the Knuth-Bendix order.

Lemma 14. *Let $\mathcal{F}$ be a signature and w be a weight function over $\mathcal{F}$ such that for all $f \in \mathcal{F}$, $\mathsf{w}(f) > 0$. Then there exists a constant M such that for all terms t: $|t| \leqslant \mathsf{w}(t) \leqslant M \cdot |t|$.*

We specialise the definition of function problems given in Section 2 to the present context. Let $R(w, v)$ denote a binary relation that can be decided by a (nondeterministic) TM M. We say the function problem F_R associated with R can be solved in linear space, if it can be solved in space linear in $|w|$.

Theorem 15. *Let $\mathcal{R}$ be a TRS, let $\mathcal{R}$ be compatible with a positively weighted KBO, and let R denote the relation computed by $\mathcal{R}$. Then the function problem F_R can be solved on a (nondeterministic) TM in linear space. Moreover if $\mathcal{R}$ is confluent then the function computed by $\mathcal{R}$ can be computed in linear space on a deterministic TM.*

Proof. Assume R is computed by the 7-tuple $(\Sigma, \mathcal{F}, \mathtt{inp}, \mathtt{out}, \mathsf{f}, \mathcal{R}, \mathsf{No})$. Note that any rewrite step can be simulated (in linear space) on a TM. Indeed, pattern matching, binary subtraction, binary addition, character erasing, and character replacements can be done in linear space. The translation `out` provides a mapping from terms to the alphabet Σ. Thus it is easy to see how to define a suitable (nondeterministic) TM $\mathsf{M} = (Q, \Sigma, \Gamma, \triangleright, \sqcup, \Delta, \mathrm{s}, \mathrm{t}, \mathrm{r})$ such that any derivation over $\mathcal{R}$ can be simulated by M. Let s, t be terms such that $s \to^* t$. By definition of KBO, $\mathsf{weight}(s) \geqslant \mathsf{weight}(t)$ and thus due to Lemma 14 $\mathsf{weight}(t) = \mathrm{O}(|s|)$. Without loss of generality we can assume that the mapping `out` is linear in the size, that is, for any term t we have $|\mathtt{out}(t)| = \mathrm{O}(|t|)$. Thus the mentioned simulation can be performed by a TM running in space $\mathrm{O}(|\mathtt{out}(s)|)$. Hence the function problem F_R is computable by a TM in linear space. If in addition $\mathcal{R}$ is confluent, M is deterministic. □

Theorem 16. *Let M be a (nondeterministic) TM that solves in linear space the function problem associated with relation R. Then R is computable by a TRS $\mathcal{R}$ that is compatible with a positively weighted KBO. Moreover if M is deterministic, the TRS $\mathcal{R}$ is confluent.*

Proof. Without loss of generality we can assume that M runs in space $k \cdot n + l$ on input w, when $n = |w|$. Set $S(n) = k \cdot n + l$. By assumption M decides the relation R, hence it halts on any input. Thus we can assume M runs in time $2^{S(n)}$. Due to Theorem 13 there exists a TRS $\mathcal{R}'$ that computes the relation R up-to S-padding. Furthermore, as S is an affine function, Lemma 12 is applicable to transform $\mathcal{R}'$ to a TRS $\mathcal{R}$ that computes R (without padding). Moreover Theorem 13 and Lemma 12 guarantee that $\mathcal{R}$ is compatible with a positively weighted KBO. □

By Theorem 15 and 16 we obtain that positively weighted Knuth-Bendix orders characterises precisely the class of function problems FNLINSPACE on arbitrary TRS and the class FLINSPACE on confluent TRS. More precisely, if we restrict our attention to languages, positively weighted Knuth Bendix orders exactly captures (LINSPACE) NLINSPACE on (confluent) TRSs.

7 Characterising Polynomial Space via KBO

In this section, we establish a complete characterisation of the complexity class PSPACE. For that it suffices to consider unary TRS. However, we have to make use of a more liberal use of padding function than in Section 6 above.

Our results rely on the insight that KBO induces exactly exponential derivational complexity on unary TRS. This is due to Hofbauer (compare [8, 6]).

Proposition 17. *For all unary TRS $\mathcal{R}$ compatible with KBO $\mathsf{dc}_{\mathcal{R}}(n) = 2^{\mathrm{O}(n)}$ holds. Furthermore there exists a unary TRS $\mathcal{R}$ compatible with KBO such that $2^{\Omega(n)} = \mathsf{dc}_{\mathcal{R}}(n)$.*

Let $R(w, v)$ denote a binary relation that can be decided by a (nondeterministic) TM M. We say the function problem F_R associated with R can be solved in polynomial space, if it can be solved in space polynomial in $|w|$.

Theorem 18. *Let $\mathcal{R}$ be a unary TRS, let $\mathcal{R}$ be compatible with a KBO, and let R denote the relation computed by $\mathcal{R}$ up-to polynomial padding. Then the function problem F_R is solvable on a (nondeterministic) TM in polynomial space. Moreover if $\mathcal{R}$ is confluent then the function computed by $\mathcal{R}$ can be computed in polynomial space on a deterministic TM.*

Proof. Assume $R(w, v)$ is computed by the 7-tuple $(\Sigma, \mathcal{F}, \mathtt{inp}, \mathtt{out}, \mathsf{f}, \mathcal{R}, \mathsf{No})$ up-to the padding function P, where P is a polynomial. Without loss of generality we assume that the translations $\mathtt{inp}$, $\mathtt{out}$ are linear in size. Let s, t be terms. Consider a derivation $D\colon s \to^* t$, such that $\mathsf{dh}(s) = n$. Due to Proposition 17, there is a constant $c > 2$ such that $s \to^n t$ implies that $n \leqslant 2^{c\cdot(|s|+1)}$. Define $d = \max(2, \max(|r| : l \to r \in \mathcal{R}))$. By assumption $\mathcal{R}$ is compatible with a KBO. Hence $\mathcal{R}$ is non-duplicating. Thus each rewrite step increases the size by at most by d. We obtain:

$$|t| \leqslant |s| + d \cdot 2^{c\cdot(|s|+1)} \leqslant 2^{|s|+2} + 2^{c\cdot(|s|+1)+d} \leqslant 2^{(c+d+2)(|s|+1)} \,.$$

In the first inequality, we apply the definition of the constant d and in the third, we make use of the inequalities $2 \leqslant |s| + 2$ and $2 \leqslant d + c \cdot (|s| + 1)$. We set $e = c + d + 2$ and obtain $|t| \leqslant 2^{e\cdot(|s|+1)}$. Let i denote the special symbol. We decompose the term t as follows: $t = \mathsf{i}^{j_0} g_1 \mathsf{i}^{j_1} g_2 \mathsf{i}^{j_2} \cdots g_{i_m} \mathsf{i}^{j_m}$, where for all k: $g_k \in \mathcal{F} \setminus \{\mathsf{i}\}$ and thus m is precisely $|t|_\sigma$. By the above estimation on the size of t, we conclude that $j_k \leqslant 2^{e\cdot(|s|+1)}$ for all k $(0 \leqslant k \leqslant m)$. In order to simulate the derivation D on a TM we need to make sure that we can encode each term succinctly. For that we represent expressions i^{j_k} by $(j_k)_2$, where the

latter denotes the binary representation of the number j_k. Hence we define $\ulcorner t \urcorner$ of t as follows: $\ulcorner t \urcorner := (j_0)_2 g_1 (j_1)_2 g_2 (j_2)_2 \cdots g_{i_m} (j_m)_2$. First, observe that $|(j_k)_2| \leqslant \log_2(2^{e\cdot(|s|+1)}) = e \cdot (|s|+1)$. Second, due to Lemma 11, there exist constants ℓ, ℓ', such that $m = |t|_\sigma \leqslant \ell \cdot \mathsf{weight}(t) \leqslant \ell \cdot \mathsf{weight}(s) \leqslant \ell' \cdot |s|$. (Recall that we have $s \to^* t$). In summary, we obtain the following estimate for the size of $\ulcorner t \urcorner$:

$$\begin{aligned} |\ulcorner t \urcorner| &\leqslant m + (m+1) \cdot (e \cdot (|s|+1)) \\ &\leqslant (m+1) \cdot (e \cdot (|s|+2)) \leqslant (\ell' \cdot |s| + 1) \cdot (e \cdot (|s|+2)) \,. \end{aligned}$$

Set $Q(n) := (\ell' \cdot n + 1) \cdot (e \cdot n + 2)$, which is a quadratic polynomial in n. Consider now some word w in the alphabet Σ. The initial term $\mathsf{f}(\mathtt{inp}(w) \sqcap^{P(|w|)})$ has size $P(|w|) + |w| + 2$, which is a polynomial in $|w|$. Applying the above equations we see that for all t such that $\mathsf{f}(\mathtt{inp}(w) \sqcap^{P(|x|)}) \to^* t$, the size of $\ulcorner t \urcorner$ is bounded by $Q(P(|w|) + |w| + 2)$, that is, polynomially bounded. As indicated above, each rewrite step can be simulated in linear space. Thus, any derivation $\mathsf{f}(\mathtt{inp}(w) \sqcap^{P(|w|)}) \to^* t$ can be simulated on a (nondeterministic) TM in space polynomial in $|w|$.

Hence the function problem F_R is computable by a TM in polynomial space. If in addition $\mathcal{R}$ is confluent, M can be assumed to be deterministic. □

Theorem 19. *Let M be a (nondeterministic) TM that solves in polynomial space the function problem associated with relation R. Then R is computable up-to polynomial padding by a unary TRS $\mathcal{R}$ such that $\mathcal{R}$ is compatible with a KBO. Moreover if M is deterministic, the TRS $\mathcal{R}$ is confluent.*

Proof. Observe that given such a machine M, there is a polynomial P such that M works in space $P(n)$ and time $2^{P(n)}$. Then the theorem follows by a straightforward application of Theorem 13. □

As a consequence of Theorem 18 and 19 we obtain that Knuth Bendix orders characterises precisely the class of function problems FPSPACE on unary TRS if computation up-to polynomial padding is considered. For languages, we conclude that KBO over unary TRS exactly captures PSPACE if polynomial padding is allowed. Observe that it is insignificant, whether the considered TRS is confluent or not.

8 Characterising Exponential Space via KBO

In this section, we establish a complete characterisation of the complexity class ESPACE.

Below we introduce a (non-standard) extension of Knuth Bendix orders. Recall the restriction to admissible weight functions in Definition 8. The following examples shows that we cannot dispense with this restriction (compare [9]).

Example 20. Let $\sqsupset$ denote the binary relation induced by Definition 8 induced by a non-admissible weight function and precedence $\succ$. Let $f, g \in \mathcal{F}$ such that $\mathsf{w}(f) = \mathsf{w}(g) = 0$ and $f \succ g$. Then $\sqsupset$ gives rise to the following infinite decent: $f(x) \sqsupset g(f(x)) \sqsupset g(g(f(x))) \sqsupset g(g(g(f(x)))) \ldots$

Let t be a term, let $f_1, \ldots, f_k \in \mathcal{F}$ denote function symbols of equal arity n. Suppose g is a fresh function symbol of arity n not in $\mathcal{F}$. We write $t[g \leftarrow f_1, \ldots, f_k]$ to denote the result of replacing all occurrences of f_i in t by g.

Definition 21. *Let* w *a (not necessarily admissible) weight function,* $\succ$ *a precedence and* $\succ_{\mathsf{kbo}}$ *be the order induced by* w *and* $\succ$ *according to Definition 8 but using only Cases 1, 2b, and 2c. Let* $f_1, \ldots, f_k \in \mathcal{F}$ *be unary symbols such that* $\mathsf{w}(f_i) = 0$ *and let* $\mathsf{w}(g) = 0$ *for a fresh symbol* g*. Suppose* $g \succ h$ *for any* $h \in \mathcal{F} \setminus \{f_1, \ldots, f_k\}$*. Then* $\succ_{\mathsf{kbo}}$ *is said to be an* enriched Knuth Bendix order *if in addition for any terms* s *and* t*,* $s \succ_{\mathsf{kbo}} t$ *implies that* $s[g \leftarrow f_1, \ldots, f_k] \succcurlyeq_{\mathsf{kbo}} t[g \leftarrow f_1, \ldots, f_k]$ *(Property (†)). Here* $\succcurlyeq_{\mathsf{kbo}}$ *denote the quasi-order induced by* $\succ_{\mathsf{kbo}}$*. The equivalence relation induced by* $\succcurlyeq_{\mathsf{kbo}}$ *is denoted as* $\simeq_{\mathsf{kbo}}$*.*

It is easy to see that an enriched KBO $\succ_{\mathsf{kbo}}$ is closed under context and substitutions. For the later it is essential, that Case 2a in the standard definition of KBO is omitted. Moreover, the property (†) in Definition 21 guarantees that $\succ_{\mathsf{kbo}}$ is well-founded. Let $\mathcal{R}$ be a TRS. We say $\mathcal{R}$ is compatible with an enriched KBO $\succ_{\mathsf{kbo}}$. if $\mathcal{R} \subseteq \succ_{\mathsf{kbo}}$. Note that compatibility of $\mathcal{R}$ with an enriched KBO implies termination of $\mathcal{R}$.

Remark 22. It is perhaps important to emphasise that $\succ_{\mathsf{kbo}}$ does not admit the subterm property. Hence an enriched KBO is not a simplification order.

We obtain the following generalisation of Proposition 11 to enriched KBOs.

Lemma 23. *Let* $\mathcal{R}$ *be a TRS over a signature* $\mathcal{F}$ *that consists only of nullary or unary function symbols and let* $\mathcal{R}$ *be compatible to* $\succ_{\mathsf{kbo}}$ *as defined above. Then* $\mathsf{dc}_{\mathcal{R}}(n) = 2^{2^{\mathrm{O}(n)}}$*. Furthermore, the size of any term of a derivation is bounded by* $2^{\mathrm{O}(n)}$*.*

Let $R(w, v)$ denote a binary relation that can be decided by a (nondeterministic) TM M. We say the function problem F_R associated with R can be solved in exponential space, if it can be solved in space $2^{\mathrm{O}(|w|)}$. The next theorem follows directly from Lemma 23.

Theorem 24. *Let* $\mathcal{R}$ *be a unary TRS, let* $\mathcal{R}$ *be compatible with an enriched KBO, and let* R *denote the relation computed by* $\mathcal{R}$*. Then the function problem* F_R *associated with* R *is solvable on a (nondeterministic) TM in exponential space. Moreover if* $\mathcal{R}$ *is confluent then the function computed by* $\mathcal{R}$ *can be computed in polynomial space on a deterministic TM.*

Theorem 25. *Let* M *be a (nondeterministic) TM that solves the function problem* F_R *associated with relation* R *in exponential space. Then* R *is computable by a unary TRS* $\mathcal{R}$ *such that* $\mathcal{R}$ *is compatible with an enriched KBO. Moreover if* M *is deterministic, the TRS* $\mathcal{R}$ *is confluent.*

Proof. Let $S(n) = 2^{k \cdot n}$ denote the bounding function of M. In proof, we assume M is nondeterminstic and we utilise the step by step simulation of a nondeterminstic TM $\mathsf{M} = (Q, \Sigma, \Gamma, \triangleright, \sqcup, \Delta, \mathrm{s}, \mathrm{t}, \mathrm{r})$ as seen in Lemma 7. In the following

we define a 7-tuple $(\Sigma, \mathcal{F}, \mathtt{inp}', \mathtt{out}, \mathsf{h}, \mathcal{R}, \mathsf{No})$ that computes R. Essentially $\mathcal{R}$ will extend the simulating TRS $\mathcal{R}_{\mathsf{M}}$ as define in Lemma 7, but we need to prepare the input suitably. Let $\overline{\Sigma}$ denotes a (disjoint) copy of the alphabet Σ. We define two translations $\mathtt{inp}$, $\mathtt{inp}'$ from Σ to $\mathcal{F}(\Sigma)$ and $\mathcal{F}(\overline{\Sigma})$, respectively. Set $\mathtt{inp}\colon a \in \Sigma \mapsto a \in \mathcal{F}(\Sigma)$ and $\mathtt{inp}'\colon a \in \Sigma \mapsto \overline{a} \in \mathcal{F}(\overline{\Sigma})$. Consider the following auxiliary TRS $\mathcal{R}_{\mathsf{dec}}$ that implements the binary decrement, where $\top$, $\bot$ denotes 1 and 0, respectively.

$$\begin{aligned}
\top(\bullet) &\to \bot(\mathsf{0} \sqcup (\bullet)) & \bot(\bullet) &\to \top'(\mathsf{0} \sqcup (\bullet)) \\
\bot(\top'(x)) &\to \top'(\top(x)) & \top(\mathsf{0}(x)) &\to \bot(\mathsf{0} \sqcup (\mathsf{0}(x))) \\
\bot(\mathsf{0}(x)) &\to \top'(\mathsf{0} \sqcup (\mathsf{0}(x))) & \top(\top'(x)) &\to \bot(\top(x)) \\
\mathsf{init}(\mathsf{0}(x)) &\to \mathsf{0}(x) & \mathsf{init}(\top'(x)) &\to \mathsf{init}(x)\ .
\end{aligned}$$

An inductive argument shows that $\mathsf{init}(\top^n(\bullet)) \to^* (\mathsf{0}\sqcup)^{2^n-1}(\bullet)$ holds for all $n \geqslant 1$. Furthermore we make use of the following TRS $\mathcal{R}'_{\mathsf{cpy}}$:

$$\begin{aligned}
\mathsf{cpy}_2(\bullet) &\to \bullet & \mathsf{cpy}_2(\overline{f}(x)) &\to f(\top^k(\mathsf{cpy}_2(x))) \\
\triangleright'(\mathsf{0}(x)) &\to \mathsf{0}(\triangleright'(x)) & \triangleright'(g(x)) &\to \triangleright(\mathrm{s}_g(x)) \\
\mathsf{h}(x) &\to \triangleright'(\mathsf{f}_{\leftarrow\mathsf{0}}(\mathsf{init}(\mathsf{f}_{\to\top}(\mathsf{cpy}_2(x))))) & \mathsf{init}(h(x)) &\to h(\mathsf{init}(x))
\end{aligned}$$

where $f \in \mathcal{F}(\Sigma)$, $\overline{f} \in \mathcal{F}(\overline{\Sigma})$ its copy, $g, h \in \mathcal{F}(\Sigma)$, and the symbols $\mathsf{f}_{\leftarrow\mathsf{0}}$, $\mathsf{f}_{\to\top}$ are defined as in Examples 2, 3. Let w be a word over Σ, let $n = |w|$, and let $\mathtt{inp}(w) = g_1 g_2 \cdots g_n$. Then it is not difficult to see that any derivation over $\mathcal{R}_{\mathsf{dec}} \cup \mathcal{R}'_{\mathsf{cpy}}$ has the following shape:

$$\begin{aligned}
\mathsf{h}(\mathtt{inp}'(w)) &\to^* \triangleright'(\mathsf{f}_{\leftarrow\mathsf{0}}(\mathtt{inp}(w)\mathsf{init}(\top^{k\cdot n}))) \\
&\to^* \triangleright'(\mathsf{0}^{2^{k\cdot n}-1}\mathtt{inp}(w)\sqcup^{2^{k\cdot n}-1})) \\
&\to^* \mathsf{0}^{2^{k\cdot n}-1} \triangleright \mathrm{s}_{g_1} g_2 \cdots g_n \sqcup^{2^{k\cdot n}-1}\ .
\end{aligned}$$

We set $\mathcal{R} := \mathcal{R}_{\mathsf{dec}} \cup \mathcal{R}'_{\mathsf{cpy}} \cup \mathcal{R}_{\mathsf{f}_{\to\top}} \cup \mathcal{R}_{\mathsf{f}_{\leftarrow\mathsf{0}}}$. It follows from Lemma 7 that the 7-tuple $(\Sigma, \mathcal{F}, \mathtt{inp}', \mathtt{out}, \mathsf{h}, \mathcal{R}, \mathsf{No})$ computes the relation R. Finally, it is not difficult to see how to define a suitable enriched KBO $\succ_{\mathsf{kbo}}$, such that $\mathcal{R} \subseteq \succ_{\mathsf{kbo}}$. □

As a consequence of Theorem 24 and 25 we obtain that enriched Knuth Bendix orders characterises precisely the class of function problems FESPACE on unary TRS. For languages, we conclude that enriched KBO over unary TRS exactly captures ESPACE. Observe that it is insignificant, whether the considered TRS is confluent or not.

9 Conclusion

In this paper we study three different space complexity classes: LINSPACE, PSPACE, and ESPACE, which are commonly believed to be distinct. We give complete characterisations of these classes exploiting TRSs, compatible with

KBOs. To capture LINSPACE, we consider confluent, positively weighted KBOs. To capture PSPACE, we consider unary TRS, where we allow for padding of the input. And to capture ESPACE, we make use of enriched KBO. For all considered complexity classes, we show similar results for the corresponding nondeterministic classes. In this case the TRS $\mathcal{R}$ need no longer be confluent.

Furthermore we have also characterised the associated classes of function problems for the (nondeterministic) space complexity classes considered. This (technical) result is of some importance as it overcomes the problem that complexity classes are languages, while TRSs compute functions.

References

1. Avanzini, M., Moser, G.: Complexity Analysis by Rewriting. In: Proc. of 9th FLOPS. LNCS, vol. 4989, pp. 130–146. Springer Verlag (2008)
2. Avanzini, M., Moser, G.: Dependency pairs and polynomial path orders. In: Proc. of 20th RTA. pp. 48–62 (2009)
3. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
4. Bonfante, G., Cichon, A., Marion, J.Y., Touzet, H.: Algorithms with polynomial interpretation termination proof. JFP 11(1), 33–53 (2001)
5. Bonfante, G., Marion, J.Y., Moyen, J.Y.: Quasi-interpretations and small space bounds. In: Proc. of 16th RTA. pp. 150–164 (2005)
6. Hofbauer, D.: Termination Proofs and Derivation Lengths in Term Rewriting Systems. Ph.D. thesis, Technische Universität Berlin (1992)
7. Hofbauer, D.: Termination proofs with multiset path orderings imply primitive recursive derivation lengths. TCS 105(1), 129–140 (1992)
8. Hofbauer, D., Lautemann, C.: Termination Proofs and the Length of Derivation. In: Proc. 3rd RTA. LNCS, vol. 355, pp. 167–177 (1989)
9. Knuth, D., Bendix, P.: Simple word problems in universal algebras. In: Leech, J. (ed.) Computational problems in abstract algebra. Pergamon (1970)
10. Koprowski, A., Waldmann, J.: Arctic Termination . . . Below Zero. In: Proc. of 19th RTA. LNCS, vol. 5117, pp. 202–216. Springer Verlag (2008)
11. Kozen, D.: Theory of Computation. Springer Verlag (2006)
12. Lepper, I.: Derivation lengths and order types of Knuth-Bendix orders. TCS 269, 433–450 (2001)
13. Marion, J.Y.: Analysing the Implicit Complexity of Programs. IC 183, 2–18 (2003)
14. Moser, G.: Derivational complexity of Knuth Bendix orders revisited. In: Proc. 13th LPAR. LNCS, vol. 4246, pp. 75–89 (2006)
15. Moser, G., Schnabl, A.: The Derivational Complexity Induced by the Dependency Pair Method. In: Proc. of 20th RTA. LNCS, vol. 5595, pp. 255–260. Springer Verlag (2009)
16. Moser, G., Schnabl, A., Waldmann, J.: Complexity Analysis of Term Rewriting Based on Matrix and Context Dependent Interpretations. In: Proc. of 28th FSTTICS. pp. 304–315. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2008)
17. TeReSe: Term Rewriting Systems, Cambridge Tracks in Theoretical Computer Science, vol. 55. Cambridge University Press (2003)
18. Weiermann, A.: Termination proofs by lexicographic path orderings yield multiply recursive derivation lengths. Theoretical Computer Science 139(1), 355–362 (1995)