LETTER

A Query System for Texts with Macros

Keehang KWON[†a)], *Nonmember*, **Dae-Seong KANG**[††], *Member, and* **Jinsoo KIM**[†††], *Nonmember*

SUMMARY We propose a query language based on extended regular expressions. This language extends texts with text-generating macros. These macros make it possible to define languages in a compressed, elegant way. This paper also extends queries with linear implications and additive (classical) conjunctions. To be precise, it allows goals of the form $D \multimap G$ and $G_1 \& G_2$ where D is a text or a macro and G is a query. The first goal is solved by adding D to the current text and then solving G. This goal is flexible in controlling the current text dynamically. The second goal is solved by solving both G_1 and G_2 from the current text. This goal is particularly useful for internet search.

key words: *text search, internet, information retrieval, linear logic*

1. Introduction

Regular expressions [1] have gained much interest as a query language for text search. While regular expressions provide multiplicative ANDs and additive ORs, they lack additive (or classical) conjunctions which are integral to internet search [2]. Lacking such connectives, regular expressions do not have a simple way to express the queries of this kind: find a web page which contains both the word *abc* and the word *bca*.

On the other hand, classical logic has gained much interest as a query language for internet search (*e.g.*, Google, Yahoo). However, classical logic lacks the notion of resources, *i.e.*, multiplicative conjunctions. Lacking such connectives, classical logic does not have an easy way to express queries of this kind: find a web page which contains at least two occurrences of the word *abbc*.

This paper introduces a superset of these two languages, which is based on propositional intuitionistic linear logic [3], [4]. This logic extends regular expressions by new forms of queries: an implication goal of the form $D \multimap G$ and the expression of the form $G_1 \& G_2$ where D is a text and G is a goal. The former one has the following intended semantics: the text D is intended to be added to the current context in the course of proving G. This expression thus supports the idea of *local* texts. This expression is particularly useful for limiting the search space for internet search. The latter expression has the following intended semantics: prove both G_1 and G_2 from the current text. This expression thus supports the idea of additive ANDs.

This logic also proposes a new form of macros for texts: both a text of the form $G \multimap D$ and a text of the form $!D$ are allowed where D is a text and G is a goal. The former one has the following intended semantics: the text G can be transformed to D. This linear logic expression thus supports the idea of (dynamic) macros. This expression is particularly useful for replacing a text with another text. The latter expression has the following intended semantics: D is reusable. This expression thus supports the idea of unbounded texts.

In this paper we present the syntax and semantics of this extended language, show some examples of its use, and study the interactions among the newly added constructs.

The remainder of this paper is structured as follows. We describe the language and an algorithm in the next section. In Sect. 3, we present some examples. In Sect. 4, we describe an improved algorithm. Section 5 concludes the paper.

2. The Language

The extended language to be considered is described by G- and D-formulas given by the syntax rules below:

$$G ::= A \mid \top \mid G_1 \otimes G_2 \mid G_1 \& G_2 \mid G_1 \oplus G_2 \mid D \multimap G \mid D \Rightarrow G$$
$$D ::= A \mid G \multimap D$$

In the rules above, A represents a word, *i.e.*, a sequence of alphabets. A D-formula is either a word (A) or a macro ($G \multimap D$). The goal $\top$ consumes arbitrary resources. We sometimes write GG in place of $G \otimes G$. The above language is based on propositional intuitionistic logic and is a subset of Lolli [4].

In the transition system to be considered, G-formulas will function as queries and the sequence of D-formulas will constitute a context. For this reason, we refer to a G-formula as a goal, to the D-formulas as a context. Our language is an extension to regular expression with the following difference: additive conjunctions, new scoping constructs are added in G-formulas, and macros are allowed in D-formulas.

We will present an operational semantics for this language.

Definition 2.1. Let G be a query and let Γ and Δ sequences

Manuscript received July 4, 2007.

[†]The author is with the Department of Computer Engineering, DongA University, Busan, 607–714, Korea.

[††]The author is with the Department of Electronics Engineering, DongA University, Busan, 607–714, Korea.

[†††]The author is with the Department of Computer Science, Konkuk University, Korea.

a) E-mail: khkwon@dau.ac.kr

DOI: 10.1093/ietisy/e91–d.1.145

of D-formulas. Then $\Gamma; \Delta \vdash G$ – the notion of proving G from $\Gamma; \Delta$ – is defined as follows:

Operational Semantics

$$\frac{}{\Gamma; \Delta \vdash \top}\top\mathcal{R}$$

$$\frac{\Gamma; \Delta_1 \vdash B_1 \quad \Gamma; \Delta_2 \vdash B_2}{\Gamma; \Delta_1, \Delta_2 \vdash B_1 \otimes B_2}\otimes\mathcal{R}$$

$$\frac{\Gamma, B; \Delta \vdash C}{\Gamma; \Delta \vdash B \Rightarrow C}\Rightarrow\mathcal{R}$$

$$\frac{\Gamma; \Delta, B \vdash C}{\Gamma; \Delta \vdash B \multimap C}\multimap\mathcal{R}$$

$$\frac{\Gamma; \Delta_1 \vdash G_1 \ldots \Gamma; \Delta_m \vdash G_n}{\Gamma; \Delta_1, \ldots, \Delta_n, G_1 \multimap \ldots \multimap G_n \multimap A \vdash A}\text{Bchain}$$

$$\frac{\Gamma; \Delta \vdash B_1 \quad \Gamma; \Delta \vdash B_2}{\Gamma; \Delta \vdash B_1 \,\&\, B_2}\&\mathcal{R}$$

$$\frac{\Gamma; \Delta \vdash B_1}{\Gamma; \Delta \vdash B_1 \oplus B_2}\quad\frac{\Gamma; \Delta \vdash B_2}{\Gamma; \Delta \vdash B_1 \oplus B_2}\oplus\mathcal{R}$$

In the above rules, Γ represents the unbounded context which can be used arbitrary times. Δ represents the bounded context which can be used only once. In proving $G_1 \otimes G_2$ from $\Gamma; \Delta$, it splits Δ into two disjoint parts. The symbol $\multimap$ provides a scoping mechanisms: it allows for the augmentation of the text in the course of proving a goal.

3. Examples

This section describes the use of our language. An example of the use of this construct is provided by the following query which searches *file* for the letters *apple* and *piano*.

egrep nil (*file* $\multimap(\top apple \top \,\&\, \top piano \top)$).

This expression contains the new scoping constructs. Now, based on the semantics that we have discussed informally above, we see that the file *file* is not available at the top-level but becomes available when evaluating the expression $(\top apple \top \,\&\, \top piano \top)$.

Our language in Sect. 2 permits a new macro to be added to the bounded part of *file* before solving the query:

egrep file $((\top kill \top \multimap love) \multimap \top kill \top)$.

This expression replaces – if there is one – an occurrence of the word *kill* by *love* in *file* before solving $\top kill \top$.

The following example permits a new macro to be added to the unbounded part of *file* before solving the query:

egrep file $((madonna \multimap ma) \Rightarrow \top ma \top)$.

This expression replaces for arbitrary times the occurrences of the word *madonna* by *ma* in *file* before solving $\top ma \top$. This is useful for user customization.

It is interesting to note that our language is a useful tool for limiting the search space for internet search. For this, let us assume that we reserve the keyword *internet* for World Wide Web. This is shown below:

egrep internet
www.ucla.edu$\otimes$*www.berkeley.edu*$\multimap \top AI \top AI \top$

This last expression searches the two UC campuses for web pages that contain at least two occurrences of the word *AI*. It is easily observed that there is no way to express this query in classical logic or in regular expressions.

4. An Improved Algorithm

We have described an algorithm in which nondeterminism is present in several places. In particular, there is a choice concerning which way the text is split in the $\otimes$ goal.

Hodas and Miller [4] dealt with the goal $G_1 \otimes G_2$ by using IO-model in which each goal is associated with its input resource and output resource. The idea used here is to delay this choice of splitting as much as possible.

Following [4], we will improve the algorithm in Sect. 2 using IO model.

Definition 4.1. Let G be a query and let I, O be texts with macros (including the new constant *deletion*). Then $\langle I, G, O \rangle$ is defined as follows:

(1) $\langle I, nil, I \rangle$ holds.

(2) $\langle I, \top, O \rangle$ holds if O is a subcontext of I, *i.e.*, O results from replacing zero or more components of I with the constant *deletion*.

(3) $\langle I, A, O \rangle$ holds if A is an atom that occurs in I and O results from replacing that occurrence of c in I with the constant *deletion*.

(4) $\langle I, G_1 \& G_2, O \rangle$ holds if both $\langle I, G_1, O \rangle$ and $\langle I, G_2, O \rangle$ hold.

(5) $\langle I, G_1 \oplus G_2, O \rangle$ holds if either $\langle I, G_1, O \rangle$ or $\langle I, G_2, O \rangle$ holds.

(6) $\langle I, G_1 \otimes G_2, O \rangle$ holds if both $\langle I, G_1, M \rangle$ and $\langle M, G_2, O \rangle$ hold for some text M.

(7) $\langle I, D \multimap G_1, O \rangle$ holds if $\langle I \otimes D, G_1, O \otimes deletion \rangle$ holds.

(8) $\langle I, D \Rightarrow G_1, O \rangle$ holds if $\langle I \otimes !D, G_1, O \otimes !D \rangle$ holds.

(9) $\langle I, A, O \rangle$ holds if $G \multimap A$ occurs in I and $\langle M, G, O \rangle$ holds where M results from replacing that occurrence in I with the constant *deletion*.

In the above rules, in proving $\langle I, G_1 \otimes G_2, O \rangle$, it introduces a new existential variable M. However the choice of M requires no non-deterministic splitting using the technique of unification [5].

Let us refer to the earlier collection of evaluation rules in Sect. 2 as $DS1$ and let $DS2$ be the IO-model defined in this section. The following theorem shows the connection between $DS1$ and $DS2$. A proof of this theorem should be obvious from the discussions in [4] and can be shown using an induction on the length of evaluation.

Theorem 4.1: Let $\Gamma;\Delta$ be a context and let G be a goal. The relation $\Gamma;\Delta \vdash G$ holds in $DS1$ if and only if $\langle\Gamma;\Delta,G,O\rangle$ holds in $DS2$ where O results from replacing each word in $\Gamma;\Delta$ with the constant *deletion*.

There is a definite benefit to using the modified rule in evaluating expressions: considerable nondeterminism in search can be eliminated by this choice. This observation leads to a more viable implementation.

5. Conclusion

In this paper, we propose a language based on extended regular expressions. This language extends texts with text-generating macros. These macros make it possible to define languages in a compressed way, as shown in XML. This paper also extends queries with linear implications and additive (classical) conjunctions. The first goal supports the notion of local texts. The second goal is particularly useful for internet search.

Adding these new constructs does not degrade performance, while enhancing usability. Our ultimate interest is in a procedure for carrying out computations of the kind described above. There are still efficiency problems. One nondeterminism arises with the $\top$ construct. This requires to consume some resources during execution. Hence it is important to realize this requirement in an efficient way, as discussed in [4], [6].

Acknowledgements

This paper was supported by Dong-A University Research Fund in 2005.

References

[1] S.C. Kleene, Introduction to Metamathematics, North-Holland, Amsterdam, 1964.
[2] J. Davies, D. Fensel, and F.V. Harmelen, Towards the Semantics Web, John Wiley and Sons, 2003.
[3] J.Y. Girard, "Linear logic," Theor. Comput. Sci., vol.50, pp.1–102, 1987.
[4] J. Hodas and D. Miller, "Logic programming in a fragment of intuitionistic linear logic," J. Inf. Comput., vol.110, no.2, pp.327–365, 1994. Invited to a special issue of submission to the 1991 LICS conference.
[5] A. Martelli and U. Montanari, "An efficient unification algorithm," ACM Trans. Program. Lang. Syst., vol.4, no.2, pp.258–282, April 1982.
[6] I. Cervesato, J.S. Hodas, and F. Pfenning, "Efficient resource management for linear logic proof search," Proc. 1996 Workshop on Extensions of Logic Programming, LNAI 1050, pp.67–81, 1996.