Common Infrastructure for Finite-State Based Methods and Linguistic Descriptions

Anssi YLI-JYRÄ[1,2] and **Kimmo KOSKENNIEMI**[2] and **Krister LINDÉN**[2]
[1] CSC - Scientific Computing Ltd., P.O. Box 405, FI-02101 Espoo, Finland
[2] Department of General Linguistics, P.O. Box 9, FI-00014 University of Helsinki, Finland
{aylijyra,koskenni,klinden}@ling.helsinki.fi

Abstract

Finite-state methods have been adopted widely in computational morphology and related linguistic applications. To enable efficient development of finite-state based linguistic descriptions, these methods should be a freely available resource for academic language research and the language technology industry. The following needs can be identified: (i) a **registry** that maps the existing approaches, implementations and descriptions, (ii) managing the **incompatibilities** of the existing tools, (iii) increasing **synergy** and complementary functionality of the tools, (iv) persistent **availability** of the tools used to manipulate the archived descriptions, (v) an **archive** for free finite-state based tools and linguistic descriptions. Addressing these challenges contributes to building a common research infrastructure for advanced language technology

1. Introduction

Finite-state methods have been adopted widely in computational morphology and related tasks of natural language and speech processing, including segmentation, tokenisation, shallow parsing, name entity recognition, normalization etc. To enable efficient development of finite-state based linguistic descriptions, the underlying methods and the lexicons should be a freely available, **common and growing** resource for academic language research and the language technology industry. The idea of a common finite-state based methodology is not new, but it has not been easy to implement in large scale.

The purpose of this article is to identify some needs that are faced when we try to reach this goal, and to propose some helpful approaches to their satisfaction. These needs are discussed in Sections 2 – 6.

2. Specialized Software Registry for Finite-State Based Resources

We need to **register** finite-state tools and linguistic resources. An open registry, **FSMREG**, currently located at http://www.ling.helsinki.fi/users/aylijyra/FSMREG will be pre-populated with the entries in our local database. After the necessary extensions, this registry

- will be a locator service for commercial and non-commercial finite-state based resources
- will map file formats and algorithms that are in use in the existing resources
- will contain hypertext links to a distributed collection of examples and stub grammars that can be used as starting points for benchmarking, testing and teaching.

According to our investigations, there are at least 70 languages to which some finite-state based methods have already been applied. Moreover, we have constructed partial registry entries about a few dozen finite-state based tools (including *ALE-RA, Amore, ASTL, BELLEx3, Carmel, DFKI FSM, FIRE toolkits, FAdo, RWTH FSA, FSA (Gdansk), FSA (Groningen), fskit, fsmlibrary, GFSMNT, grmlibrary, ifsc, Intex, KIMMO, lexc, lextools, MAP (Alvey), MIT FST, MMORPH, OMAC FSM, PC-KIMMO, SFST, twolc, UCFSM, Unitex, Vaucanson, wfsc, wfst, X2MORF, xfst*).

We welcome contributions of new or corrected entries in the registry. In the future, we plan to move the registry to a collaboration environment using the wiki technology, and to present a version of the registry as a survey article or technical report.

3. Common Formats and Formalisms for Finite-State Resources

We need to manage the divergence of the existing finite-state tools. Different finite-state tools should be capable of **exchanging** various types of data: finite-state objects as well as grammar source files created in finite-state based formalisms. Currently, many finite-state based formalisms can be parsed only with a proprietary compiler. To create interoperable tools and industry standards, we need

- an open forum for reviewing idiosyncratic features of finite-state based rule formalisms
- a generic XML-based exchange format for finite-state based rule formalisms
- converters that rewrite formalisms into system specific regular expressions (For example, *xfst2fsa* (Cohen-Sygal and Wintner 2005) converts a large subset of the Xerox finite-state formalism in *xfs*, to expressions of the *FSA* utilities from Rijksuniversiteit Groningen.)
- XML-formats (such as proposed by the Vaucanson group http://www.lrde.epita.fr/cgi-bin/twiki/view/Vaucanson/XML) for exchanging small finite-state objects
- open libraries that can exchange huge finite-state objects in various binary formats

4. Complementary Modules of Finite-State Methods

We need to increase **synergy** in building new finite-state tools. Earlier, proprietary and private implementations of finite-state methods have been in-house tools for building certain natural language and speech processing applications. As a result, similar finite-state toolkits have been reimplemented several times in different places. Now that a few proprietary finite-state toolkits are available under commercial licenses, there is a great need for **complementary tools** that would help in tasks where flexibility is more important than high performance.

- We need open source tools that can be mutated and exploited more freely
- We need compilers that can be linked with different finite-state libraries:
    a. a pre-compiler for compiling linguistic descriptions into regular expressions
    b. regular expressions would be compiled by a separate program into finite-state objects

It is surprising how little the flexibility and modularity of widely available finite-state compilers has developed during the course of last 20 years. Earlier, when finite-state tools were written in the Lisp programming language, it was convenient to implement rule compilers and pre-compilers (see *e.g.* Karttunen *et al.* 1987) also in Lisp. Today, some pre-compilers for regular expressions have been implemented with XML-based techniques (Piskorski *et al.* 2002). The software package *fskit* developed by the first author employs a further pre-compiler and macro expansion method.

5. Encouraging Open Source Development of Finite-State Resources

We need an **action plan** that increases the free availability of useful finite-state based methods and descriptions. Currently, some tools for creating linguistic resources are available under incompatible or closed-term license models. The action plan would

- encourage compatibility with such research networks that build free finite-state based descriptions (including the RELEX network and OpenOffice-related projects)
- encourage the use of open source or creative commons licenses that allow linking to software covered by GNU's copyleft license as well as to proprietary software
- recognize the need for a manageable negotiation procedure in the exceptional cases where the terms of the default license is not compatible with a desirable combination
- discuss the possible need for joint copyright systems

There is a trade-off between the commercial relevance for widely spoken languages and the common good for communities of less-studied languages and the research community. This opposition has wide practical implications that make it especially complicated to build a common, standardized infrastructure for finite-state based methods and applications.

For example, the free availability of some finite-state based formalisms is perhaps not even possible due to potential patent risks. In other words, patents and proprietary programming languages are problematic from the viewpoint of persistent archiving and sustainability. They may involve risks if the value of the infrastructure of language resources is dependent on the availability of the software needed to maintain the resources.

6. Archiving

All the finite-state resources need to be archived and stored somewhere. We believe that storage is not a problem for open-source resources, but the main problem is to keep the resources maintainable and exploitable. This involves, in addition to the maintained finite-state compilers for the resources, sufficient documentation on the metadata and the used codes for each stored linguistic finite-state resource.

7. Conclusion

The better interoperability of high-end proprietary tools and freely available, sustainable tools is crucial requirement for multi-lingual language technology industry that would support diversity and development of language technology for minority languages (Yli-Jyrä 2005, Koskenniemi 2006). Open source language technology resources such as finite-state based methods and finite-state based linguistic descriptions

- create a basis for further experimental research on finite-state methods
- increase the availability of basic utilities needed in many small language technology projects
- support the development of complex applications on top of basic methods
- increase the efficiency and flexibility of commercial and academic research and development.

If the repeated investments in basic finite-state based resources could be avoided, new development efforts could concentrate on less-studied languages, research collaboration, more complex applications and the production of end-user products.

8. References

Beesley, K. R. (2004). Morphological analysis and generation: a first-step in natural language processing. In *Proceedings of the SALTMIL Workshop at LREC 2004: First Steps in Language Documentation for Minority Languages. Computational Linguistic Tools for Morphology, Lexicon and Corpus Compilation*, Lisbon, Portugal, May.

Cohen-Sygal, Y and S. Wintner. (2005). XFST2FSA: comparing two finite-state toolboxes. In *Proceedings of the ACL-2005 Workshop on Software*, Ann Arbor, MI, June 2005.

Karttunen, L., K. Koskenniemi and R. M. Kaplan. (1987). A compiler for phonological rules. In M. Dalrymple, et al. (eds). *Tools for Morphological Analysis. Center for the Study of Language and Information*. Stanford University, Palo Alto.

Koskenniemi, K. (2005). White paper: "Multilingual Europe – How to get there?" Presented in the Workshop "Machine Translation and Human Language Technologies", European Commission, Luxembourg, February.

Piskorski, J., W. Drożdżyński, O. Scherf and F. Xu. (2002). A flexible XML-based regular compiler for creation and conversion of linguistic resources. In *Proceedings of the 3rd International Conference on Language Resources an Evaluation (LREC'02)*, Las Palmas, Canary Islands, Spain.

Yli-Jyrä, A. (2005). Toward a widely usable finite-state morphology workbench for less studied languages – part I: desiderata. *Nordic Journal of African Studies*, 14(4): 479 – 491.