Compiling Linguistic Constraints into Finite State Automata

Matthieu Constant\textsuperscript{1} and Denis Maurel\textsuperscript{2}

\textsuperscript{1} Universit\’e de Marne-la-Vall\’ee, Institut Gaspard Monge, France
\textsuperscript{2} Universit\’e François-Rabelais de Tours, Laboratoire d’Informatique, France

Abstract. This paper deals with linguistic constraints encoded in the form of (binary) tables, generally called lexicon-grammar tables. We describe a unified method to compile sets of tables of linguistic constraints into Finite State Automata. This method has been practically implemented in the linguistic platform Unitex.

1 Motivation

Finite State Models have been intensively used in Natural Language Processing \cite{13}. Nevertheless, because of the complexity of languages, it is often more convenient for linguists to describe linguistic constraints with simpler and more ergonomic representations. For instance, simple regular expressions are sometimes used to express morphological rules \cite{6}, inflected forms of dictionaries are preferred to be written in a textual form \cite{3} and syntactic constraints depending on lexicon are represented in the form of binary matrices \cite{4}. Finite State linguistic phenomena are sometimes described with more powerful and more compact formalisms such as (weighted) context-free grammars \cite{10} and recursive transition networks\cite{5}. These representations are then compiled into Finite State Automata or Transducers in order to optimize processing.

This paper deals with linguistic constraints encoded in the form of (binary) tables made of rows and columns, generally called lexicon-grammar tables. A row of such table corresponds to the formal description of the lexical and syntactic properties accepted by a lexical item. Each column corresponds to a property. At the intersection of a row and a column, the encoded value indicates whether or not a lexical entry (row) accepts a property (column)\textsuperscript{1}. In this paper, we will describe a unified method to compile sets of tables of linguistic constraints into Finite State Automata. We will also show how it has been practically implemented in the linguistic platform Unitex \cite{11}.

2 State-of-the-Art

The first idea of combining binary matrices and automata was pointed out in \cite{7}, but the first compilation method has been found in \cite{12} and has been implemented in the linguistic platforms INTEX \cite{14} and Unitex \cite{11}. It was limited

\textsuperscript{1} Usually, symbol + stands for True and symbol - stands for False.

© Springer-Verlag Berlin Heidelberg 2006
to systems of constraints encoded in one table such as the ones in [4]. It used
hand-built parameterized reference automata, representing the sets of the pos-
sible syntactic constructions where can enter a fictive lexical entry accepting all
properties of the table. Each path is parameterized by one or several parameters
that refer to properties that correspond to syntactic constructions (e.g. Prep
Det Noun) or lexical information (e.g. if the constituent Prep accepts the lexi-
cal value in). The compilation process consists, for each lexical entry (or raw), in
resolving the parameters according to the encoding in the tables. For instance, a
false value at a given column indicates that the transitions labeled with the para-
meter associated with the column, must be removed. A true value indicates that
these transitions must be made epsilon-transitions. Then, a specific automaton
is constructed for each lexical entry. The automaton representing all described
phenomena is simply the union of all constructed automata. It is then optimized
by a deterministic minimization operation for text processing efficiency.

Several linguistic studies have shown that it is sometimes more convenient to
encode constraints of a same linguistic phenomena into systems of multiple tables
because some properties can be factorized in different tables to avoid encoding
duplication [7,1]. In this case, Roche’s compilation does not work because it does
not handle multiple tables. [8] implemented an algorithm compiling systems of
multiple tables of specific constraints. These constraints were limited to very
local constraints. Tables described the restrictions on the combinations of pairs
of lexical elements in sequences where both elements occur consecutively (or
sometimes with a grammatical word in between). For instance, for French time
expressions, sequence milieu de matin (middle of morning) is forbidden while
sequence milieu d’après-midi (middle of afternoon) is accepted. A schemata au-
tomaton is used to represent all possible patterns for a type of expressions. This
automaton also recognizes bad sequences because it does not take lexical re-
strictions into account. All forbidden sequences encoded in the tables are put in
an automaton that is then applied using the failure algorithm [9] that cuts all
forbidden paths in the schemata automaton. [2] proposed an algorithm with no
restrictions on the constraints; constraints were represented in relational systems
of tables. The algorithm consisted in directly constructing the automaton that
recognizes accepted sequences, by using a parameterized reference automaton
with parameters resembling Roche’s ones. Nevertheless, the complexity of the
construction of the parameterized automaton could grow very fast with the num-
ber of tables. For instance, it is not well adapted to Maurel’s time expressions.

In this paper, we present a unified algorithm for compiling systems of tables
of constraints with no restrictions on the type of constraints.

3 Set of Constraints and Parameterized Automaton

This section focuses on the general description of inputs of our algorithm, that
are a set of linguistic constraints and a parameterized schemata. They are re-
spectively described in section 3.1 and in section 3.2

2 Prep Det Noun stands the construction preposition determiner noun