

Typesetting XTDL Grammars and Typed Feature Structures with FS₂L^AT_EX

Ulrich Schäfer
Language Technology Lab
German Research Center for Artificial Intelligence (DFKI GmbH)
Stuhlsatzenhausweg 3
D-66123 Saarbrücken, Germany
ulrich.schaefer@dfki.de
<http://www.dfki.de/~uschaefer>

November 18, 2004

Abstract

FS₂L^AT_EX is a smart tool that generates L^AT_EX code from the XTDL grammar syntax or XML representations of typed feature structures, e.g., as generated by SProUT and JTFS [3, 1]. FS₂L^AT_EX makes it easy to insert typed feature structures into documentations, papers, theses or slides. Possible applications are (1) documentation of SProUT grammars (2) typesetting SProUT or JTFS results (3) typesetting (HPSG) feature structures in general. FS₂L^AT_EX comes in a tiny JAR archive ([fs2latex.jar](#)). The only prerequisite is a JRE 1.4 – neither SProUT nor any other Java library is required to run FS₂L^AT_EX from command line. FS₂L^AT_EX has been developed on Linux, but it should run on MS Windows as well.

1 Introduction

The command line syntax of FS₂L^AT_EX is as simple as¹

```
java -jar fs2latex.jar [-param=value]*
```

FS₂L^AT_EX acts as a filter. It reads XTDL or XML input from `stdin` and writes L^AT_EX code to `stdout`. Optionally, it can take parameters of the form `-param=value` (cf. Section 2). The input type (XTDL or XML) is detected automatically (opening angle brackets, of course).

1.1 Typesetting SProUT XTDL grammars

To typeset a SProUT rule on the fly, simply type

```
echo 'r1 :> morph & [INPUT < "This", "is", #1>, choice { [VAL #1 & \
      stupid, SEM nonsense], #2 } ] -> #2.' | java -jar fs2latex.jar > r1.tex
```

This generates L^AT_EX code for the following structure.

$$r1 :> \left[\begin{array}{l} \textit{morph} \\ \text{INPUT } \langle \text{"This", "is", } \boxed{1} \rangle \\ \text{choice } \left\{ \begin{array}{l} \text{[VAL } \boxed{1} \textit{ stupid} \\ \text{SEM } \textit{nonsense} \end{array} \right\}, \boxed{2} \end{array} \right] \rightarrow \boxed{2}.$$

¹ At DFKI-LT, the shell script `~uschaefer/bin/fs2latex [-param=value]*` can be used as well. Updates of the tool will be published at <http://www.dfki.de/~uschaefer/fs2latex/>.

To typeset a SProUT XTDL grammar file, just enter e.g.,

```
cat ne_person_name.sgr | java -jar fs2latex.jar -corefsize=tiny \
  -fontsize=scriptsize > pn.tex
```

This produces feature structures of the following form (but without line breaks within a rule, Section 3 explains how to insert these).

$$\begin{array}{l} \text{en_person_name_with_title} :> \left[\begin{array}{l} \textit{gazetteer} \\ \text{GTYPE } g_en_title \\ \text{SURFACE } \boxed{\textit{title}} \end{array} \right] \bullet @seek(\textit{unknown_person_name}) \ \& \\ \\ \left[\begin{array}{l} \text{GIVEN_NAME } \langle \boxed{\textit{name1}}, \boxed{\textit{name2}}, \boxed{\textit{name3}} \cdot \textit{*top*} \rangle \\ \text{SURNAME } \boxed{\textit{surname}} \end{array} \right] \\ \\ \rightarrow \left[\begin{array}{l} \textit{ne-person} \\ \text{TITLE } \langle \boxed{\textit{title}} \cdot \textit{*top*} \rangle \\ \text{GIVEN_NAME } \langle \boxed{\textit{name1}}, \boxed{\textit{name2}}, \boxed{\textit{name3}} \cdot \textit{*top*} \rangle \\ \text{SURNAME } \boxed{\textit{surname}} \end{array} \right] . \end{array}$$

FS₂L_AT_EX also understands and automatically detects the XTDL-XML DTD when a <RULE> or <DUMMY_RULE> element is encountered, cf. Section 7.3.

1.2 Typesetting TFS XML and SProUTput XML

XML conforming to the TFS DTD is e.g. generated by JTFS, cf. Section 7.1. To typeset an XML file, just type

```
cat de.xml | java -jar fs2latex.jar > de.tex
```

An example for on-the-fly typesetting is

```
echo '<FS type="token"><F name="ID"><FS type="id"/></F></FS>' \
  | java -jar fs2latex.jar > fs.tex
```

The following feature structures have been generated from SProUTput XML.

$$\left[\begin{array}{ll} \textit{ne-person} & \\ \text{SURFACE} & \textit{string} \\ \text{ID} & \textit{*top*} \\ \text{PREPOSITIONS} & \textit{*list*} \\ \text{DESCRIPTOR} & \textit{string} \\ \text{GIVEN_NAME} & \textit{"Peter"} \\ \text{TITLE} & \textit{string} \\ \text{SURNAME} & \left[\begin{array}{l} \textit{"Möller"} \\ \text{CSTART } \boxed{1} \textit{*top*} \\ \text{CEND } \boxed{2} \textit{*top*} \end{array} \right] \\ \text{P-POSITION} & \textit{string} \\ \text{NAME-SUFFIX} & \textit{string} \\ \text{OUTCSTART} & \boxed{1} \\ \text{OUTCEND} & \boxed{2} \\ \text{VARIANT} & \textit{"Möller"} \end{array} \right] \left[\begin{array}{ll} \textit{ne-location} & \\ \text{OUTCSTART} & \textit{*top*} \\ \text{OUTCEND} & \textit{*top*} \\ \text{VARIANT} & \textit{*top*} \\ \text{SURFACE} & \textit{string} \\ \text{ID} & \textit{*top*} \\ \text{PREPOSITIONS} & \textit{*list*} \\ \text{DESCRIPTOR} & \textit{string} \\ \text{LOCTYPE} & \textit{city} \\ \text{LOCNAME} & \textit{"München"} \end{array} \right]$$

1.3 Generated L^AT_EX code

The generated L^AT_EX file may be processed by e.g.

```
latex document.tex
```

or parts of it (e.g., a single feature structure or a SProUT rule) may be copied into another LaTeX document. In this case, the header (indicated in LaTeX comments) in document.tex should be copied only once into the header of the other document. Header generation may be disabled by specifying a `-header` parameter with a value other than `yes`.

The generated L^AT_EX code does *not* rely on the well-known `avm.sty` by Christopher Manning. The reason is that `avm.sty` requires more T_EX memory and therefore cannot be used to typeset larger feature structures without recompiling T_EX from its source code with increased stack/heap sizes. Of course, also the feature structures generated by FS2L^AT_EX have a size limit imposed by the T_EX memory settings.

2 Parameters

The parameters influence mainly the header of the generated L^AT_EX file.

Parameter name	Default value	Description
<code>fontsize</code>	<code>normalsize</code>	set font size of feature structures and types
<code>corefsize</code>	<code>footnotesize</code>	set font size of coreferences
<code>coreffont</code>	<code>rm</code>	set coreference font name
<code>typefont</code>	<code>it</code>	set type font name
<code>atomfont</code>	<code>tt</code>	set atom font name
<code>featurefont</code>	<code>rm</code>	set feature font name
<code>arraystretch</code>	<code>1.1</code>	set vertical stretch factor for feature structures
<code>header</code>	<code>yes</code>	include a full L ^A T _E X header (documentclass etc.)
<code>poster</code>	<code>no</code>	include an extended header for posters; cf. Section 5
<code>inputenc</code>	<code>latin1</code>	set input encoding for the <code>inputenc</code> package; cf. Section 6
<code>addtoheader</code>		additional L ^A T _E X header command to insert
<code>usepackage</code>		additional L ^A T _E X package to include

3 Post-Editing

If a single typed feature structure is to be typeset, a dummy XTDL rule should be created of the form `x :> [...] -> y.` and everything except the LHS part of the rule (containing the feature structure code, indicated as `% LHS ---` in L^AT_EX comments) should be removed manually. The reason for this is that a single typed feature structure is not well-formed XTDL syntax.

Manual post-Editing may also be necessary to add line and (for posters) page breaks.

In case of XTDL, a single `eqnarray*` environment is created with each rule in one line (additional lines for functional constraints), and the rule horizontally aligned at the `:>` separator (after the rule name). It may be necessary to split the `eqnarray*` block by adding

```
\end{eqnarray*}
\end{featurestruct}
\begin{featurestruct}
\begin{eqnarray*}
```

between the rules (begin of a rule is marked in a comment). It is also possible to manually add line breaks to rules exceeding the paper size horizontally, e.g., by inserting

```
\\ & &
```

(a line break and an alignment after the rule name) once or several times into the LHS of a rule. Another option is to insert additional `array` environments, e.g. within the LHS of a SProUT rule.

In case of TFS XML input, disjunctions or just sequences of feature structures are processed one after another and are put into a single `displaymath` environment without line breaks. It is possible to split multiple feature structures manually by adding additional `displaymath` environment delimiters (`\]` `\[`) between.

4 Java API

The Java API of FS2 \LaTeX basically consists of a single method, a variant of the main method that is called when starting `fs2latex.jar`, taking a String (XTDL or XML) and a Properties object containing the optional parameters as input, and returning a String containing the generated \LaTeX code. I.e., the signature² is

String `de.dfki.lt.sprout.xtdl2xml.toLaTeX.translate(String input, Properties params)`
E.g., in Java code using the JTFS package, insert the following lines to generate \LaTeX code of a `FeatureStructure` object using the `FeatureStructure.xmlFS()` and `toLaTeX.translate()` methods.

```
import de.dfki.lt.sprout.xtdl2xml.toLaTeX;
import de.dfki.lt.tfs.FeatureStructure;

FeatureStructure myFS = ...;
Properties params = new Properties();
params.setProperty("header", "no");
StringBuffer sbXML = new StringBuffer();
myFS.xmlFS(sbXML);
String latex = toLaTeX.translate( sbXML.toString(), params );
```

5 Posters

A0 posters of SProUT grammar files (XTDL or XTDL-XML) or SProUT grammar output (XML file generated by IDE/SProUT runtime) can be generated easily by specifying the `--poster=yes` parameter, e.g.,

```
cat ne_person_name.sgr | java -jar fs2latex.jar -poster=yes \
    -corefsz=tiny -fontsz=scriptsz > pn.tex && latex pn && dvi2pdf pn
```

generates first \LaTeX code and then a PDF file.

\LaTeX ing the generated code requires additional \LaTeX packages (from <http://www.ctan.org/tex-archive/macros/latex/contrib/>):

- `a0poster`
- `textpos`

For details on the installation of CTAN packages cf. Section 5 of [2].

In the DFKI-LT environment, the `~uschaefer/bin/fs2poster` command can be used to create a PDF poster with FS2 \LaTeX .

<http://www.dfki.de/~uschaefer/fs2latex/fs2poster.pdf> is a sample document generated by `fs2poster`.

²The Javadoc URL of the `toLaTeX` class is at <http://www.dfki.de/~uschaefer/fs2latex/fs2latex/toLaTeX/de/dfki/lt/sprout/xtdl2xml/toLaTeX.html>

6 Encodings

It may be necessary to convert character sets (e.g. UTF-8 as produced by SProUT runtime and IDE XML output to ISO-8859-1) before passing the generated code to the `latex` command. The GNU `recode` tool can do this, but some versions seem to have corrupted translation tables making the output document unusable. The syntax is e.g.

```
java -jar fs2latex.jar | recode utf8..latin1 > fs.tex
```

When a target encoding other than `latin1`, e.g. `latin2`, is generated, then the `inputenc` option should be specified as well, in this case `-inputenc=latin2`.

7 DTDs and XTDL BNFs

7.1 TFS-XML DTD

This is a minimal DTD at <http://sprout.dfki.de/tfsxml.dtd> for typed feature structures with sets and coreferences. Lists, e.g., are typeset in angle brackets by $\text{FS}_{\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}}$ iff typed as **cons** or **opencons** with features FIRST and REST, empty list **null**.

Surrounding DISJ elements (optional!) are handled as in the SProUTput DTD (Section 7.2). Any other XML elements in the input DTD are ignored. Multiple FSes od DISJs are listed horizontally.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- TFS XML DTD 2004 Version 2.1 -->

<!ELEMENT FS ( F )* >
<!ATTLIST FS type NMTOKEN #REQUIRED
             coref NMTOKEN #IMPLIED >

<!ELEMENT F ( FS | SET ) >
<!ATTLIST F name NMTOKEN #REQUIRED >

<!ELEMENT SET ( FS | SET )* >
<!ATTLIST SET coref NMTOKEN #IMPLIED >
```

7.2 SProUTput DTD

This is the DTD at <http://sprout.dfki.de/sproutput.dtd> for default XML output of the SProUT runtime engine. Both SET and DISJ elements are printed between braces. FS and F elements are formatted as in the TFS-XML DTD (Section 7.1). Any other XML elements (including MATCHINFO) in the input DTD are ignored. Multiple FSes and DISJs are listed horizontally.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Sproutput DTD 2004 Version 2.1 -->

<!ELEMENT SPROUTPUT ( DISJ )* >

<!ELEMENT DISJ ( MATCHINFO )+ >
<!ATTLIST DISJ id ID >

<!ELEMENT MATCHINFO ( FS ) >
<!ATTLIST MATCHINFO id ID #IMPLIED
                    rule NMTOKEN #IMPLIED
                    cstart NMTOKEN #IMPLIED
                    cend NMTOKEN #IMPLIED
                    start NMTOKEN #IMPLIED
                    end NMTOKEN #IMPLIED >

<!ELEMENT FS ( F )* >
<!ATTLIST FS type NMTOKEN #REQUIRED
             coref NMTOKEN #IMPLIED >

<!ELEMENT F ( FS | SET ) >
<!ATTLIST F name NMTOKEN #REQUIRED >

<!ELEMENT SET ( FS | SET )* >
<!ATTLIST SET coref NMTOKEN #IMPLIED >
```

7.3 XTDL-XML DTD

This is the DTD of the intermediate XML representation of the SProUT XTDL grammar syntax, as used by the SProUT grammar compiler (<http://sprout.dfki.de/xtdl.dtd>).

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- SPROUT Grammar XML DTD Version 2004
      AUTHOR : {krieger,scherf,uschaefer,witold}@dfki.de
      VERSION: 2.1
      DATE:    2003-12-19
      NOTES: CFS is restricted additionally by the parser in the following way.
             Negation and seek are allowed only on the toplevel and only on the LHS.
             Sets are allowed only as values of features, but not on the toplevel.
             COLLECT with no type attribute is only allowed on the LHS of a rule,
             COLLECT with type set or list is only allowed on the RHS of a
             rule. Multiple TYPE, COREF, COLLECT or FS elements are not
             allowed as direct children of a CFS element. -->

<!ENTITY % rvalue "DISJ | CONCAT | N-TIMES | RANGE | STAR | PLUS |
                  ZERO-ONE | CFS | SEEK" >

<!ELEMENT SPROUT-GRAMMAR ( RULES ) >

<!ELEMENT RULES ( RULE | DUMMY_RULE )+ >

<!ELEMENT RULE ( LHS, RHS?, FNCN? ) >
<!ATTLIST RULE  name NMTOKEN #REQUIRED
                pos NMTOKENS #IMPLIED >

<!ELEMENT DUMMY_RULE ( LHS, RHS?, FNCN? ) >
<!ATTLIST DUMMY_RULE  name NMTOKEN #REQUIRED
                    pos NMTOKENS #IMPLIED >

<!ELEMENT LHS ( %rvalue; )+ >

<!ELEMENT RHS ( CFS ) >

<!ELEMENT FNCN ( FN )+ >

<!ELEMENT FN ( CFS )+ >
<!ATTLIST FN  name NMTOKEN #REQUIRED
              coref NMTOKEN #IMPLIED
              pos NMTOKENS #IMPLIED >

<!ELEMENT DISJ ( ( %rvalue; ), ( %rvalue; )+ ) >

<!ELEMENT CONCAT ( ( %rvalue; ), ( %rvalue; )+ ) >

<!ELEMENT N-TIMES ( %rvalue; ) >
<!ATTLIST N-TIMES num NMTOKEN #REQUIRED >

<!ELEMENT RANGE ( %rvalue; ) >
<!ATTLIST RANGE start NMTOKEN #REQUIRED
                end NMTOKEN #REQUIRED >
```

```

<!ELEMENT STAR ( %rvalue; ) >

<!ELEMENT PLUS ( %rvalue; ) >

<!ELEMENT ZERO-ONE ( %rvalue; ) >

<!ELEMENT SEEK ( CFS? ) >
<!ATTLIST SEEK name NMTOKEN #REQUIRED
                pos NMTOKENS #IMPLIED >

<!ELEMENT SET ( CFS | SET )+ >

<!ELEMENT CFS ( TYPE | FS | SET | COREF | COLLECT )* >

<!ELEMENT FS ( F )* >
<!ATTLIST FS neg (true) #IMPLIED >

<!ELEMENT F ( CFS ) >
<!ATTLIST F name NMTOKEN #REQUIRED
            pos NMTOKENS #IMPLIED >

<!ELEMENT TYPE ( #PCDATA ) >
<!ATTLIST TYPE neg (true) #IMPLIED
              pos NMTOKENS #IMPLIED >

<!ELEMENT COREF ( #PCDATA ) >
<!ATTLIST COREF dct NMTOKEN #IMPLIED
                pos NMTOKENS #IMPLIED >

<!ELEMENT COLLECT ( #PCDATA ) >
<!ATTLIST COLLECT type (list|set) #IMPLIED >

```

7.4 BNF for XTDL

This is the ‘normative’ BNF generated by JJDoc from the JavaCC grammar for XTDL (as of 2004-09-03). There is also a simplified BNF in the SProUT documentation, cf. <http://sprout.dfki.de>. There are additional restrictions like that negation of @seek is not allowed, i.e., if necessary, it must be inserted \neg manually into the generated code (cf. Section 3). Another restriction is that only a single type is permitted as type of a typed feature structure.

Non-Terminals

```
Input          ::= Rules
Rules          ::= Rule ( Rule )*
Rule          ::= <IDENTIFIER> ( ":"> | ":"/ ) Conc "->" toplevelCfsRhs
              ( "," ( "where" | "WHERE" | "Where" ) Functions )? "."
Functions     ::= Function ( "," Function )*
Function      ::= ( Coref "=" )? <IDENTIFIER> "(" ( FnArgs )? ")"
FnArgs       ::= embeddedCfs ( "," embeddedCfs )*
Seek         ::= ("@seek" | "@SEEK") "(" <IDENTIFIER> ")"
toplevelCfsLhs ::= toplevelTerm ( "&" toplevelTerm )*
toplevelCfsRhs ::= toplevelTermRhs ( "&" toplevelTermRhs )*
embeddedCfs   ::= embeddedTerm ( "&" embeddedTerm )*
toplevelTermRhs ::= ( ( FeatureStructure | List | Type | Coref | Collect ) )
toplevelTerm  ::= ( ( "~" )? ( FeatureStructure | List | Type | Coref | Collect | Seek ) )
embeddedTerm  ::= ( FeatureStructure | List | Type | Coref | Collect | Set )
Coref        ::= "#" ( <IDENTIFIER> | <NUMBER> )
Collect      ::= ( "%" ( ( CoName ) | "{" ( CoName "}" ) | "<" ( CoName ">" ) ) )
CoName       ::= ( <IDENTIFIER> | <NUMBER> )
Type         ::= ( <IDENTIFIER> | "..." | <STRING> )
FeatureStructure ::= <"[" AttrValPair ( "," AttrValPair )* "]">
AttrValPair  ::= <IDENTIFIER> embeddedCfs
Set          ::= ( "{" embeddedCfs ( "," embeddedCfs )* "}" )
List         ::= "<" ( embeddedCfs ( "," embeddedCfs )* ( "." embeddedCfs )? )? ">"
Conc        ::= Disj ( Disj )*
Disj        ::= Kleene ( "|" Kleene )*
Kleene      ::= Element ( ( "+" | "*" | "?" | ( "{" <NUMBER> ( "," <NUMBER> )? "}" ) ) )?
Element     ::= ( toplevelCfsLhs | "(" Conc ")" )
```

Tokens

```
< NUMBER:      (< DIGIT >)+>
< STRING:     "\" ( ~[\""] | "\\\" | "\\\" )* \" >
< IDENTIFIER:
  < LETTER >|
  ((< LETTER >|< DIGIT >|< SPECIAL >)+ < SPECIAL > (< LETTER >|< DIGIT >|< SPECIAL >)* |
  ((< LETTER >|< DIGIT >|< SPECIAL >)* < SPECIAL > (< LETTER >|< DIGIT >|< SPECIAL >)+) |
  ((< LETTER >|< SPECIAL >)+ < DIGIT > (< LETTER >|< DIGIT >|< SPECIAL >)* ) |
  ((< LETTER >|< DIGIT >|< SPECIAL >)* < DIGIT > (< LETTER >|< SPECIAL >)+)
  >
< #LETTER:    [ "A" - "Z", "a" - "z", "\u0370" - "\uFFFF" ]>
< #DIGIT:     [ "0" - "9" ] >
< #SPECIAL:   [ "_", "+", "-", "*", "?" ]>
}
```

8 Copyright

© 2004 German Research Center for Artificial Intelligence (DFKI GmbH) Language Technology Lab, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany.

Downloads, DTD and documentation are available from <http://www.dfki.de/~uschaefer/fs2latex/>

BibTeX entry for this manual:

```
@manual{author = {Ulrich Sch{"a}fer},
        title = {Typesetting {XTDL} Grammars and Typed
                Feature Structures with {FS2LaTeX}},
        url = {http://www.dfki.de/~uschaefer/fs2latex/fs2latexman.pdf},
        organization = {{DFKI GmbH}, Language Technology Lab},
        address = {Saarbr{"u}cken, Germany},
        year = 2004,
        month = 11}
}
```

References

- [1] Witold Drożdżyński, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. Shallow processing with unification and typed feature structures — foundations and applications. *Künstliche Intelligenz*, 2004(1):17–23, 2004. URL: http://www.kuenstliche-intelligenz.de/archiv/2004_1/sprout-web.pdf. 1
- [2] Peter Flynn. *A beginner's introduction to typesetting with L^AT_EX*. Silmaril Consultants, Textual Therapy Division, November 2003. URL: <ftp://dante.ctan.org/text-archive/documentation/beginlatex/beginlatex.a4.pdf>. 4
- [3] Hans-Ulrich Krieger, Witold Drożdżyński, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. A bag of useful techniques for unification-based finite-state transducers. In *Proceedings of KONVENS-2004*, pages 105–112, Vienna, Austria, 9 2004. URL: <http://www.dfki.de/dfkibib/publications/docs/sproutKONVENS2004.pdf>. 1
- [4] Walter Schmidt, Jörg Knappen, Hubert Partl, and Irene Hyna. *L^AT_EX₂ε-Kurzbeschreibung*. Dante e.V., April 2003. URL: <ftp://dante.ctan.org/text-archive/documentation/lshort/german/l2kurz.pdf>.