Building a XMG Compiler

Simon Petitjean

Contents

1 — Writing Language Bricks Files

1.1 Introduction

To build a XMG Compiler that fits your needs, the first step is to define the language you will use.

The MetaGrammatical language is defined in a modular way, as a combination of elementary parts of language, called Language Bricks. Each language Brick is described into a file, usually named *BrickName.xmg*

A Language Brick is a set of grammar rules, that describe all the structures allowed in the language. A grammar rule is the couple of a name (called non terminal symbol) and a structure, which is a sequence of non terminals and terminals (words of the language).

The syntax used to write Language Bricks is very close to the Yacc syntax. It begins with the definition of the language terminals and non terminals, and end with the rewriting rules, each of these sections being followed by '%%' .

1.2 Terminals

A XMG terminal is basically a keyword of the language. The terminal *term* is declared in the following way:

```
%token term
```

1.3 Non Terminals

The non terminal *NonTerminal* is declared in the following way:

```
%type NonTerminal NT
```

1.4 Externs

There is a special type of non terminals we call Extern non terminals. Their role is to be the connections between Language Bricks. An extern must appear only in the right side of rules.

```
%ext NonTerminal NT
```

1.5 Rules

A rule associates a non terminal to the structures it describes. The left part of the rule must be a non terminal, and the right part a sequence of terminals and non teminals. A rule has the following form:

```
Node : node Props Feats ;
```

The right part of a rule can be a disjunction of sequences:

```
Node : node Props Feats | node id Props Feats ;
```

1.6 Punctuation

Punctuations are parsed in the same way as terminals (they are just special terminals), but do not have to be declared. They come in the right part of rules with simple quotes. Here is an example of rules including punctuation:

```
Expr : '{' Expr '}' ;
```

1.7 An example Language Brick

Here is an example of a Language Brick:

```
%type Stmt Stmt
%type Call Call
%type ids_coma ids_coma
%type Dot Dot
%type Var Var
%ext DimStmt DimStmt

%token id

%%

  Stmt :        DimStmt | Stmt ';' Stmt | Stmt '|' Stmt | Var '=' Call | Call
                | Var '=' Dot | Var '=' Var | '{' Stmt '}' ;
  Dot :         id '.' Var ;
  Call :        id '[' ids_coma ']' | id '[' ']';
  ids_coma :    id ',' ids_coma | id;
  Var :         id | '?' id;
%%
```

2 — Combining Bricks

2.1 Connect

When all the Language Bricks files needed by the user are written, Language Bricks can be created and combined to build the MetaGrammatical Language. A Language Brick is represented by a python object called BrickGrammar, which takes as a parameter a Language file.

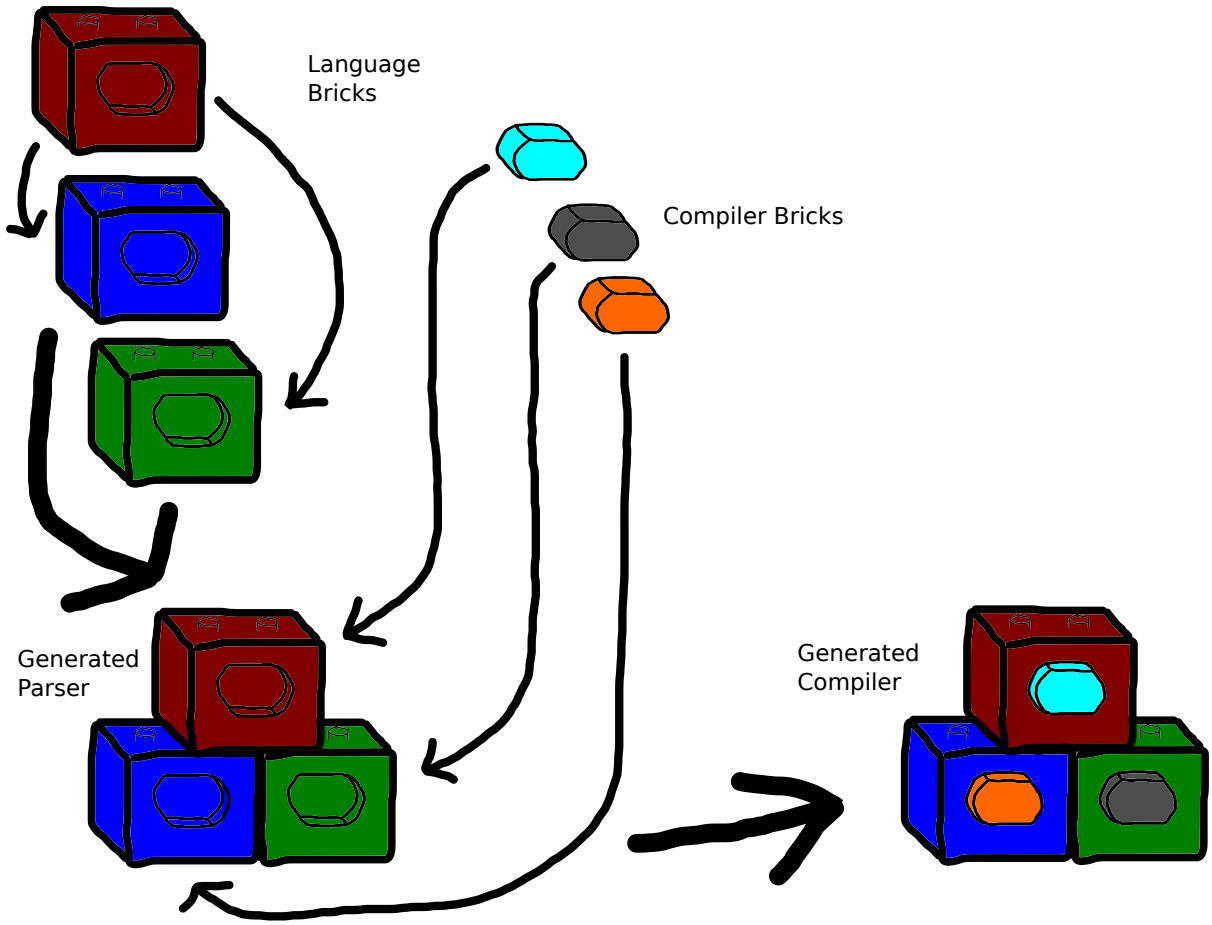
```
DeclLang = BrickGrammar('xmg-mg.xmg', prefix='XMG')
```

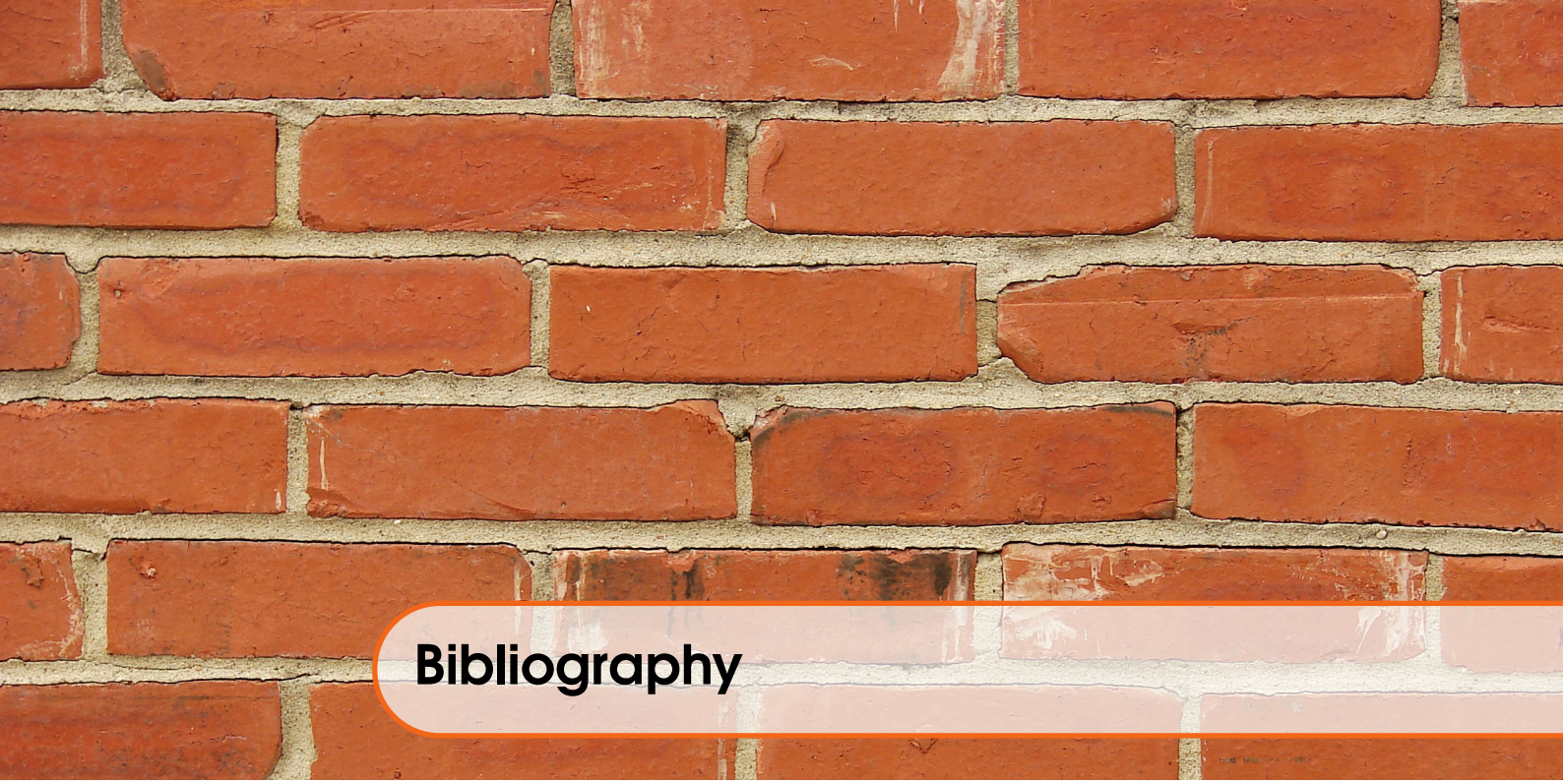
3 — Compiler Bricks

3.1 Patterns

4 — Building the Compiler

4.1 Putting it all together





Bibliography

Index