Basic Concepts

Compiler generator

A **compiler generator** is a program that translates a specification into a compiler for the programming language described in the specification.

Specification file

A **specification file** specify the words and the sentences accepted in a language. Here is an example:

```
// Words

Tokens

   if = 'if';
   then = 'then';
   else = 'else';
   assign = '=';
   eq = '==';
   neq = '!=';
   id = ['a'..'z']+;
```

```
// Sentences

Productions

  instr
    = {if} if exp then instr |
      {if_else} if exp then instr else instr |
      {assign} id assign exp;

  exp
    = id op id;

  op
    = {equal} eq |
      {unequal} neq;
```

Grammar

We use a **grammar** to define the accepted sentences of a language. As seen in the previous figure, a grammar is a set of **productions** of the form :

symbol = symbol … symbol

Derivation

To show that a sentence is in the language described by a grammar, we can perform a **derivation**.

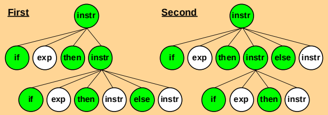
Using the previous grammar, here are two derivations of the sentence **if a==b then if b==c then d=a==c else d=a==b** :

```
First

instr = if exp then instr
instr = if a==b then instr
instr = if a==b then if exp then instr else instr
instr = if a==b then if b==c then instr else instr
instr = if a==b then if b==c then d=exp else instr
instr = if a==b then if b==c then d=a==c else instr
instr = if a==b then if b==c then d=a==c else d=exp
instr = if a==b then if b==c then d=a==c else d=a==b
```

```
Second

instr = if exp then instr else instr
instr = if a==b then instr else instr
instr = if a==b then if exp then instr else instr
...
```

Parse tree

A ***parse tree*** is made by connecting each symbol in a derivation to the one from which it was derived. Here are two incomplete parse trees for the previous derivations :



Ambiguous grammar

A grammar is ambiguous if it can derive a sentence with two parse trees. In this case, we said that the grammar contains a **conflict**.

SableCC

SableCC

SableCC is an object-oriented framework that generates compilers (and interpreters) in the Java programming language.

Build a compiler using sableCC

Here are the **steps to build a compiler** using SableCC :

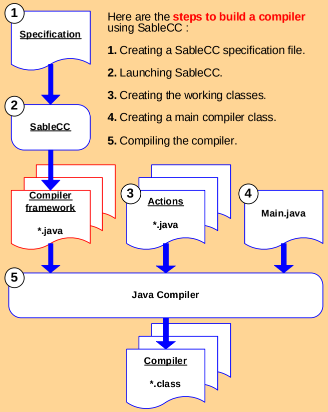
1. Creating a SableCC specification file.

2. Launching SableCC.

3. Creating the working classes.

4. Creating a main compiler class.

5. Compiling the compiler.



Old SableCC conflict message

When SableCC processes a conflictual grammar, it issues a conflict message and exits. Here is the **old conflict message** issued by SableCC for the grammar wrote in the previous section :

```
shift/reduce conflict in state [stack: TIf PExp TThen PInstr *] on TElse in
{
   [ PInstr = TIf PExp TThen PInstr * TElse PInstr ] (shift),
   [ PInstr = TIf PExp TThen PInstr * ] followed by TElse (reduce)
}
```

This messages tells us that after the parser has successfully parsed the sequence of tokens and productions [if exp then instr], if it sees an 'else' token, it cannot decide whether it is:

a) (**shift**) parsing an if/then/else construct,

```
    if exp then
       instr
    else
       instr
```

or

b) (**reduce**) if it has seen a complete if/then instruction followed by the upcoming 'else'.

```
    if exp then
       instr

    else ...
```

Unintuitive message

If the parser interprets the prefix **[if exp then exp]** as **[instr]**, it is somehow saying that there should exist a program of the form **[instr else ...]**. Yet, no such program can exist, given the grammar wrote in the previous section.

We have modified the way SableCC informs us that a grammar contains a conflict.

UQÀM

Université du Québec à Montréal

Visual Conflict Debugger Eclipse Plug-in

for the
SableCC compiler generator

http://sablecc.org

by

Sébastien Adam

sebastien@sablecc.org

Eclipse

Eclipse Plug-ins

In Eclipse, we have created a new perspective containing a grammar editor and a view that shows our visual debugger.

In addition, we have integrated SableCC to allow the users to process his grammar. Hence, the users can write, process, and fix his grammar in the same environment.

SableCC Plug-in

Until now, SableCC was useable only from the command line. We have **integrated SableCC in Eclipse** by creating a new workbench menu and a new editor context menu. We have implemented some menu items that trigger the functionalities of SableCC.

Visual Debugger Plug-in

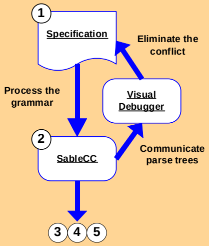
We have **integrated our visual debugger in Eclipse** by implementing a view, two tree viewers, a label provider, and a content provider.

Visual Debugger

Visual Debugger

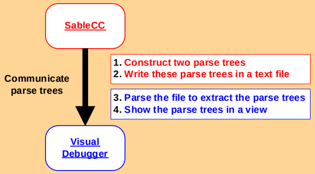
Our visual debugger aims to simplify the task of understanding and fixing conflicts in a grammar.

The figure below illustrates where the visual debugger appears in the **steps to build a compiler** using SableCC.



Parse trees specification file

By now, when SableCC processes a conflictual grammar, it provides two parse trees to the visual debugger by using a text file.



1. Construct two parse trees
2. Write these parse trees in a text file

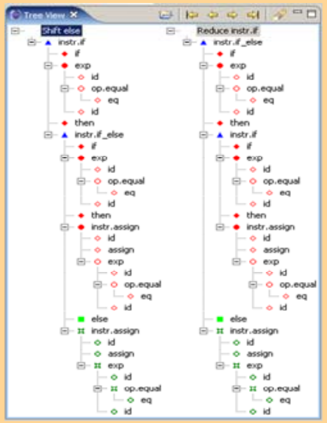
3. Parse the file to extract the parse trees
4. Show the parse trees in a view

By using a text file, **we separate the debugger tool from SableCC** itself. This choice allows for other parser generators to use our visual debugger.

Parse-Tree Viewer

Our visual debugger **illustrates a conflict** in a grammar by showing the parse trees obtained from the compiler generator.

Here is the **parse-tree viewer** that illustrates the conflict found in the grammar seen before.



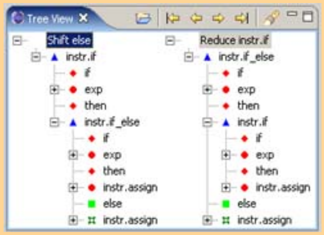
The table below lists the various **types of node** found in the parse-tree viewer.

Node Type	Production Representation	Token Representation
Parent	▲	none
Main Prefix	◆	◆
Prefix	◇	◇
Lookahead	none	■
Suffix	⌗	◇

Default Expansion

The debugger provides a **default tree expansion** which only shows the important elements involved in a conflict.

This expansion can be performed by expanding parent nodes, as shown in the figure below.



Link Back to the Grammar

Once a conflict has been understood, the user must go back to editing the grammar to eliminate the conflict. To simplify this, our visual debugger provides a contextual menu that allows the user to **navigate from a symbol in a parse-tree viewer** to the related line of the input grammar file.

Equivalent Nodes Matching

Double clicking on any main prefix, prefix, or lookahead node of a tree causes the appropriate expansion of the other tree in order to **highlight the related equivalent node**.

This should help the user to better identify similarities and differences between the two parse trees to understand the conflict.

References

[1] Gagnon, Etienne. SableCC, An Object-Oriented Compiler Framework. Master's thesis, McGill University, Montreal Quebec, March 1998.

[2] SableCC: A Compiler Generator. http://sablecc.org/.

[3] Eclipse: An Open Extensible IDE. http://eclipse.org/.

[4] IBM Corporation. Eclipse Platform Technical Overview, 2003. World-Wide Web page URL: http://www.eclipse.org/articles/index.html.