Russian Academy of Sciences

Program Systems Institute

Research Center for Multiprocessor Systems
Programming Automation Laboratory

The Refal Plus Programming System

1. Implementation of the Refal plus programming language based on array list representation.

Array representation of Refal expressions has a number of advantages including effective evaluation of the Refal expression length during the program execution process and construction of subexpression by its length and position in the Refal expression. Effective implementation of these operations permits to develop and use a **new approach to an effective implementation of syntax identification** in the Refal language.

2. Strict compilation of Refal programs into an imperative language. The scheme of a strict compilation of a Refal program into an abstract imperative language which could be easily transformed into any target platform by using the back-end module has been implemented.

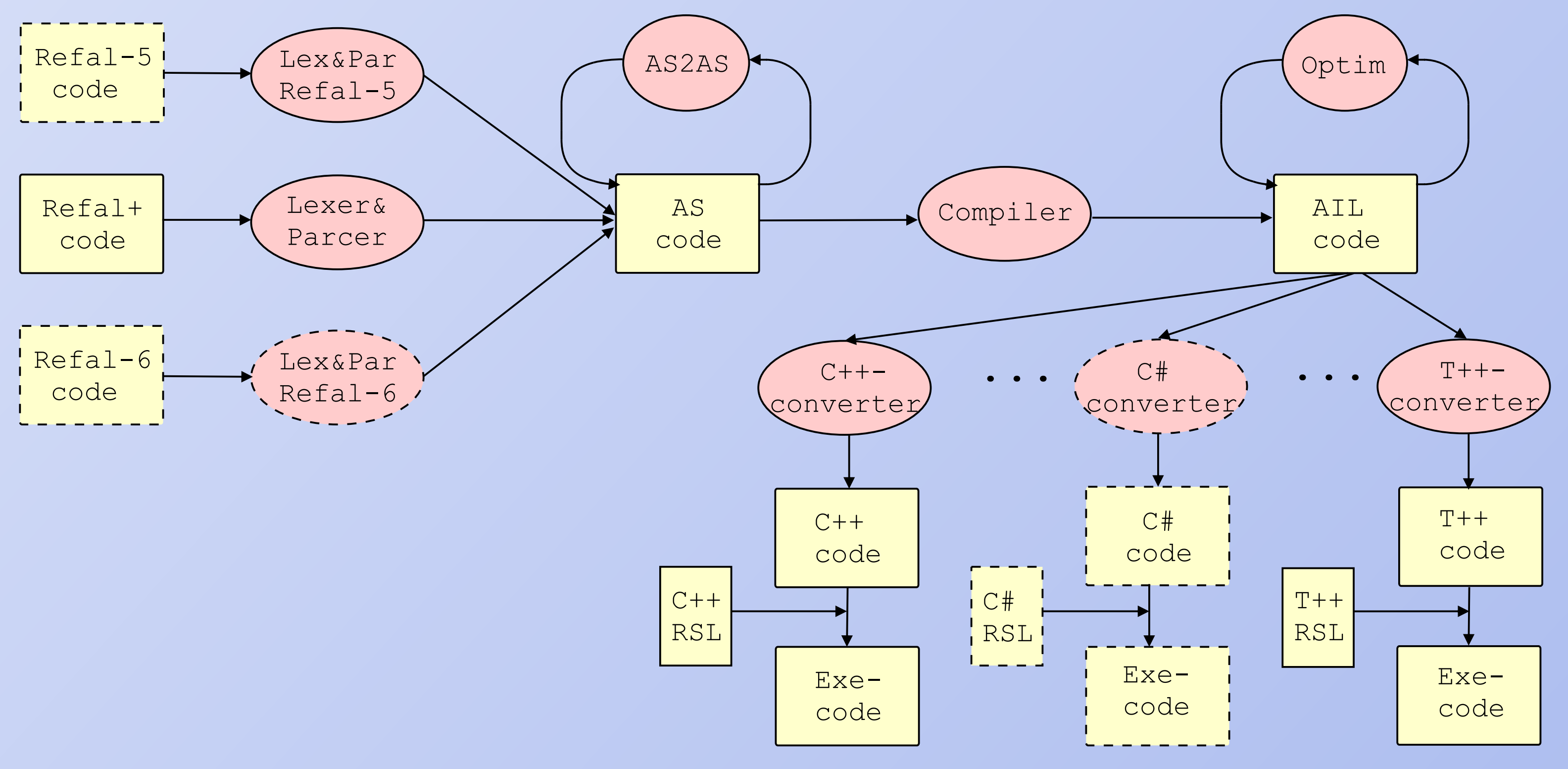
3. Openness, flexibility, and considerable modularity. The system is implemented as a number of separate modules with clearly described interfaces. The modules are available in source codes. The compiler of the system is written on Refal plus; variants of runtime support libraries for different platforms are written on **high-level languages (C++, Java, T++). Moreover, this approach provides a high level of system portability.**

4. Support of possible extensions. **T**he module structure of the system ensures easy functionality changing without modification of the whole system and/or a considerable part of its code.

Consequently, we have a possibility:

- to add different (and optimizing) transformations of Refal programs in abstract syntax format (AS2AS-transformers);
- to add different (and optimizing) transformations of the result of compilation in abstract imperative language format (Optim modules);
- to include different Refal dialects into the system (those which exist and will appear in the future) using the possibility to add different front-end modules (Lexer&Parser);
- to support different target architectures and platforms in the system, implement different versions of back-end modules and corresponding runtime support libraries and, due to this, to implement the system on the base of both traditional imperative languages (C++, Java, C#) and the languages supporting parallel execution of programs in multiprocessor systems (T++, OpenTS).

In order to implement the described approach the compilation process of a Refal Plus program is divided into four main parts:



- The Lexer&Parser module tranforms a Refal Plus program into the intermediate Refal- oriented language called Abstract Syntax (AS-code). Abstract syntax has tools for an appropriate representation of particular features of all Refal dialects.
- The compiler module transforms AS-code into programs on an abstract imperative language.
- The converter module generates output text on an imperative language.
- Generation of the executable module on the basis of a corresponding runtime support library.

Example of translation of a Refal Plus program into C++

Refal Plus program

```
$Use StdIO
Main=<PrintLN 'Hello!'>;
```

C++ program

```
#include <rf_core.hh>
#include <refal/StdIO.hh>
namespace refal
{
using namespace rfrt;
namespace hello
{
static const Expr
_c_0=Char::create_expr('Hello!');
RF_FUNC (Main, (), (RF_RES _v_res1))
RF_CALL (StdIO::PrintLN,(_c_0),());
_v_res1=empty;
RF_END
}
rfrt::Entry rf_entry (hello::Main);
}
```

ADDRESS

Research Center for
Multiprocessor Systems
Program Systems Institute
Russian Academy of Sciences

Pereslavl-Zalessky
Yaroslavl Region
Russia, 152020
Tel/Fax: **+7 (08535) 980**64
E-mail: abram@botik.ru
Web-site: http://www.botik.ru/PSI/RCMS