
Mengfei Ren(mr3258)
Yun Feng(yf2294)
Chang Hu(ch2950)
Jiacheng Chen(jc3940)
Yu Kang(yk2540)
1. Introduction

This manual describes the iChemi language. The iChemi programming language is an effective programming language for balancing chemical equations. Except common operations (declaration, condition flow, loop, block, function call, etc.), it provides an easy way to express the chemical formula. And with the build-in function that serves to balance equations.

2. Lexical Conventions

2.1 Notation

Through the document, nonterminals are in italics and terminals are bold format. Regular expression-like constructs are used to simplify grammar presentation.

- r* means the pattern r may appear zero or more times.
- r+ means the r may appear one or more times.
- r? means r may appear zero or once.
- r1 | r2 denotes an option between two patterns.
- r1 r2 denotes an option between two patterns, r1 r2 denotes r1 followed by r2.

2.2 Comments

We use # to mark comments and we only support single line comment. If the length of comments exceeds one line, remember to put # at the beginning of each line.

2.3 Identifiers

An identifier consists of a letter followed by other letters, digits and ".". Identifiers are case sensitive, so "left" and "Left" are different identifiers. The length of an identifier is not limited.

2.4 Keywords

The reserved keywords in iChemi are:

    int  double  string  bool
    true  false  if  else
2.5 Constants

2.5.1 Integer Constants

Integer constants should be decimal and consists of a sequence of numbers without a decimal point. All integers should be unsigned.

Accept: 45, 0
Not accept: -1, +12, 1.5

2.5.2 Double Floating-point Constants

A double floating-point constant consists of a sequence of numbers and a decimal point. Before the decimal point is the integer part and the decimal part is after the point. The integer part could be omit. Examples of valid and invalid cases are listed:

Accept: 0.5
.234
10.55
Not accept: 1
1.
.1.1
2..5

2.5.3 String Constants

String constants are demarcated by double quote characters, for instance “iChemi”. Escape characters are supported.

<table>
<thead>
<tr>
<th>Character Name</th>
<th>Escape Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newline</td>
<td>\n</td>
</tr>
<tr>
<td>Horizontal tab</td>
<td>\t</td>
</tr>
<tr>
<td>Backslash</td>
<td>\ \</td>
</tr>
</tbody>
</table>
2.5.4 Boolean Constants

Boolean constants consist of the keywords true and false. For example:
bool b = true

2.6 Operators and Punctuations

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>plus/minus/multiple/divide</td>
</tr>
<tr>
<td>-</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td></td>
</tr>
<tr>
<td>/</td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than/equal/not equal</td>
</tr>
<tr>
<td>=</td>
<td>less than/greater than equal</td>
</tr>
<tr>
<td>!</td>
<td>to/less than equal to</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>logical and/logical or</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>Put equation element inside</td>
</tr>
<tr>
<td>[ ]</td>
<td>comma(separate expressions)</td>
</tr>
<tr>
<td>;</td>
<td>end of expression</td>
</tr>
<tr>
<td>()</td>
<td>separators</td>
</tr>
</tbody>
</table>

3. Meaning of Identifiers

3.1 Scoping

In iChemi global and local variables are both supported. A variable defined inside a block is local, and cannot be called outside the block. For example:

fun int Add(int a, int b)
{
    int c = 0;
    return a+b+c;
}
int b = c; #This is illegal

3.2 Object Types

In iChemi, five fundamental types of objects are supported. They are as following:
- int
- double
- string
- bool
Integer type
In iChemi the only supported integer type is int which can store 32-bits worth of data. This data type is signed.

Double type
In iChemi the only supported double type is double which can store 64-bits worth of data. This data type is signed.

String type
In iChemi we provide string type, which means a string of unlimited length could be supported. However, the length may be limited because of the amount of computer resources.

Bool type
In iChemi bool type is supported. Bool type could only take a value of either true or false. However, bool type is very useful and helpful.

Fun type
All the functions users write could be supported as fun type which could return a value and be transferred to other variables.

Equation
We provide this build-in type for balancing chemical equations.

4. Operator Conversions

4.1 Operations

<table>
<thead>
<tr>
<th>Tokens</th>
<th>Operators</th>
<th>Class</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>* /</td>
<td>Multiplicative</td>
<td>Binary</td>
<td>L</td>
</tr>
<tr>
<td>+ -</td>
<td>Additive</td>
<td>Binary</td>
<td>L</td>
</tr>
<tr>
<td>&lt; &lt;= =&gt; &gt;</td>
<td>Relational comparisons</td>
<td>Binary</td>
<td>L</td>
</tr>
<tr>
<td>== !=</td>
<td>Equality comparisons</td>
<td>Binary</td>
<td>L</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>Logical AND</td>
<td>Binary</td>
<td>L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Logical OR</td>
</tr>
<tr>
<td>!</td>
<td>Logical NOT</td>
<td>Unary</td>
<td>R</td>
</tr>
<tr>
<td>=</td>
<td>Assignment</td>
<td>Binary</td>
<td>R</td>
</tr>
<tr>
<td>,</td>
<td>Comma</td>
<td>Binary</td>
<td>L</td>
</tr>
</tbody>
</table>
4.2 Primary Expressions

Identifiers, constants, strings. The type of the expressions depends on identifier, constant or string.

4.3 Arithmetic operators

In iChemi, arithmetic operators are +, -, *, and /. + means addition, - means subtraction, * means multiplication and / means division. All of them are binary and left associative. It requires that their operands must be of the same primitive types, and the result will be of the same type.

4.4 Comparative operators

In iChemi, comparative operators are > (greater than), < (less than), >= (greater than or equal to), <= (less than or equal to), != (not equal) and == (equal). All of them are binary operators and left-associative. It requires that their operands must be of the same primitive types. The return value is a boolean value indicates the predicate.

4.5 Logical operators

Logical operators in iChemi include && (logical and), || (logical or) and ! (logical not). && and || are binary operators and left-associative. They take two operands of type boolean, and return a boolean value. ! is unary and appears on the left side of the operand. They type of the operand must be of type boolean and the return type is also a boolean value.

4.6 Assignment operators

iChemi’s assignment operator is =. It's a binary operator and right-associative. The left operand must be a legal left value, and the right operand must be an expression. When an assignment is taken, the value of the expression on the right is assigned to the left value, and the new value of the left value is returned.

4.7 [ ] #equation element

In iChemi, we put the equation elements in this operator, separate them by comma: [element1], [element2], [element3]
5. Grammar

5.1 Program Definition

A program in the iChemi language is simply a sequence of declarations and statements which are executed in order.

Program -> Declaration* Statement*

5.2 Declarations

All variables must be declared before they can be used. However, variable declarations can be made at any point in a program. Variables become usable after the end of the semi-colon of the statement in which it contained.

In iChemi, declarations consists of variable declarations and function declarations.

Declaration -> Variable Declaration*  
| Function Declaration*

5.2.1 Variable Declaration

Variable declarations are used to set variables equal to computed values. The type used in a declaration must match with the type that the expression on the right hand side of the assignment operator returns. A variable declaration consists of the following grammar.

Variable Declaration -> datatype Identifier SEMICOLON  
| datatype Identifier EQUAL Expression SEMICOLON

For example:

int a;
double a=5.3*2;

Declare variables continually is forbidden.

For example:

int a, b, c;
int a=0; b =1;
int a=b=0;
5.2.2 Function Declaration

iChemi Language supports functions. Function Declaration is defined below:

Function Declaration -> fun returnType Identifier ((datatype variable)*
   |(datatype variable COMMA)+ datatype variable) Block

For example:

fun int Add (int a, int b)
{
   return a+b;
}

5.3 Statement

Statements in iChemi contains declarations, assignments, conditional statement, loop statement, function calls, return statement, null statement, etc.

Statement ->
   Block*
   |Declaration*
   |Assignment*
   |IF statement*
   |WHILE statement*
   |RETURN
   |Function Call*
   |NULL statement*

5.3.1 Expressions

An expression statement is composed of primary statements with a semicolon at the end of the line. It is used for binary operations.

5.3.2 Assignments

Assignment is to give a variable a value.

Assignment -> Identifier = Expression SEMICOLON
5.3.2 If Statements

iChemi supports two kinds of if-else statements:

If Statement ->
If (Boolean conditions) Statement
|If (Boolean condition) Statement else Statement

No else if (or “if nest”) is allowed. So if more conditions are needed, use more blocks. For example:

if(a=true) {}
else
{
  if(b = true) {}
}

5.3.3 While Statements

C programmers are often tolerate with for statement, as they have to type in verbose statement. While, in iChemi, to avoid verbose, we just use while as the loop statement.

While Statement -> while (condition) Statement

5.3.4 Return Statements

A compute function returns a value to the caller through return statements.

Return Statement -> return (Expression)? SEMICOLON

5.3.5 NULL Statements

In iChemi, if a line contains just a semicolon, that means a null statement and has no meaning.

5.3.6 Blocks

A block is a container contains statements. The variables in different blocks have different “level property” and might lead to variable scoping problems.

Block -> LEFTBRACE Statement* RIGHTBRACE
5.3.8 Function Calls

In the statement, functions declared before could be called by using function name. iChemi provides two build-in function read(), print() as I/O operation. The build-in class Equation also has a function balance() to balance chemical equations, and function add() to set equation reactant and products properties, which will be described in Chapter 6.

6. Equation Class

6.1 Equation Declarations

iChemi has a build-in class Equation, which is used for chemical equation balancing. To declare a chemical equation is just like general function declarations in a format:

```
Equation identifier
{
    Left [reactant 1], [reactant 2];
    (Right [product 1];)?
}
```

The right attribute(products) could be null in declaration. Reactant and products of a chemical equation should be write like this in iChemi:

```
(coef) ATOM1^no_ATOM2^no
```

coeff is the coefficient of the reactant/product, and the atoms of a material should be separated by ‘.’. And each atom should be followed by its number appears in the material. For example, in the chemical equation $2\text{H}_2 + \text{O}_2 = 2\text{H}_2\text{O}$, we could write

```
Equation water {
    Left [(2)H^2], [O^2];
    Right [(2)H^2_O];
}
```

6.2 Usage of Equation

After declare an equation, we could use add() function to add new materials to the
equation. For example, e.Right.add(product1).

We could also call balance() function to balance the chemical equation in the statement like e.balance().

References