IconicLisp: A Visual Lisp Programming

Muhammed S. Al-Mulhem
Information and Computer Science Department
King Fahd University of Petroleum and Minerals
Dhahran, 31261, Saudi Arabia

Abstract - This paper proposes a visual functional programming language called IconicLisp. The IconicLisp system is an interactive environment for the development of Lisp programs visually. It uses the Cons-Cell graphical notations to write visual Lisp programs. The syntax and the semantics of IconicLisp are based on the functional programming language Lisp. Once a visual program is developed, the IconicLisp system generates an equivalent textual Lisp program. The paper starts with an overview of the existing visual functional languages. Next, it gives a brief description of the cons-cell graphical notations. Finally, it shows an example of using the system and gives the conclusion.

Keywords: Functional programming, Visual functional programming.

1 Introduction

Visual programming [1] is one of the attempts to simplify the process of writing programs. Many visual programming languages have been developed for different domains and applications [2, 3].

Tinkertoy programming system [4] provides a graphical interface to Lisp. The user types a textual Lisp expression in Tinkertoy window and the system generates an equivalent iconic representation for it. Tinkertoy icons are graphical symbols with input and output ports on them through which they can be connected together with other icons to build programs. There are many icons used in Tinkertoy system such as the “print out”, “Function”, and “NIL” icons.

Visual Haskell [5] provides a visual equivalent for the Haskell functional language. It is a program visualization tool. Its graphical notation is the nested-box representation with arrows. Haskell textual programs are first translated into a textual intermediate form, which is then translated into visual Haskell. Visual Haskell syntax is based on the dataflow metaphor.

VFPE [6] uses tree structure to represent functional programs. Each expression category in a program is represented by a tree node corresponding to that category. In VFPE, each syntax element has its own unique shape and label. In addition, every syntax node has an associated data type.

Interaction Nets [7] are used as a formalism for visual functional programming. They are used to represent functional programs visually and to animate the execution of such programs by graph rewriting. In addition, they are used to reason about functional programs and to perform visual programs transformation.

This paper proposes a visual functional programming language called IconicLisp. The proposed language is used to write visual programs that are equivalent to their corresponding Lisp programs. It uses the Cons-Cell graphical notations [8] to write visual programs. IconicLisp’ syntax is based on the Scheme language [9], which is a dialect of Lisp.

2 Cons-Cell notations

Cons-Cell notations [8] consists of nodes and pointers to connect the nodes. A node is an object of two slots, called CAR and CDR. Each slot can hold or refer to any Lisp object as shown in Figure 1.

\[
\begin{array}{|c|c|}
\hline
\text{CAR} & \text{CDR} \\
\hline
\end{array}
\]

Figure 1: A node.

A list is a series of nodes, linked together so that the CDR slot of each node holds either a pointer to the next node or the empty list. For example, consider the list shown in Figure 2.

\[(A (B C) D)\]

Figure 2: A list.
Its Cons-Cell representation is shown in Figure 3.

![Figure 3: Cons-Cell representation for a list.](image)

Another example, consider the Lisp code shown in Figure 4.

```
(CONS (QUOTE A) (QUOTE B))
```

![Figure 4: A Lisp code.](image)

Its Cons-Cell representation is shown in Figure 5.

```
(CONS) (QUOTE A) (QUOTE B)
```

![Figure 5: Cons-Cell representation for a Lisp code.](image)

There are two types of Lisp objects, atom and list. IconicLisp represents these objects visually using its Cons-Cell notation. Atoms are represented as a single node and a list is represented as a list of nodes connected with pointers.

Using this simple and intuitive graphical notation is powerful enough to represent any Lisp code. In addition to its simplicity and expressiveness, it eliminates the need to write parentheses when writing Lisp code. Once a visual Lisp program is written, the IconicLisp system can generate an equivalent textual Lisp program in Scheme syntax.

### 4 IconicLisp Environment

IconicLisp system consists of two windows: the working area window and the output window. The working area window is used to write IconicLisp programs and the output window is used to display the generated equivalent textual Scheme code. Figure 7 shows the IconicLisp working area window with a visual Lisp list equivalent to the textual list (* (+ X Y) Z).

![Figure 7: The IconicLisp working area.](image)

### 3 IconicLisp language

IconicLisp uses a modified version of the Cons-Cell graphical notation to write Lisp visual programs. A node in IconicLisp consists of three slots instead of two. The first two slots are the CAR and CDR slots, which are identical to the ones found in the node of the original Cons-Cell notation. The third slot is a new one called the input port, which is used to connect the nodes together to form a list. A node in IconicLisp is shown in Figure 6.

![Figure 6: A node in IconicLisp.](image)
5 Conclusions

This paper presents a visual functional programming language called IconicLisp. It is a hybrid visual programming language based on the Scheme functional language. IconicLisp has been implemented fully in Java. It has the advantages of using one of the best and most intuitive graphical notations for Lisp programs (Cons-Cell).

6 Acknowledgment

I would like to thank KFUPM for their support. Also, I would like to thank my student K. Alzamil for implementing the system.

7 References


