Programming Language Theory

Section 1

21900628 Sechang Jang

Language is growing. Vocabulary is the set of words, and language is vocabulary and rules. A program is a list of things that will be done by computer. The term "ing" is used with a noun to indicate the term in the verb form. Programming languages are the language used to tell what to do for a computer.

If we have many words, we need to consider many things. We can express many things with many words with various rules. For example, in the language "English", the term "men" has one syllable. We can consider it as primitive. The term "women" is composed of two-syllable, primitive term "men" is used to define the term "women". A "person", which can express men or women, uses the terms "men" and "women" underneath its definition.

As we see the language is growing. We can define the new rule to support richer terms and expressions. Small language is easy to learn and straightforward. However, large languages can express many things more easily than small programming languages. There are pros and cons, the answer is the language that grows.

We need to make a space that other people define. We can't implement everything from scratch. New designs and new rules can be defined. Many programming languages started as large programming languages. Some are considered small languages now, and some are not used anymore since many people have difficulty learning everything.

The essence of language design is not designing "the right thing". User doesn't wait for the right thing, just choose the cheap and fast. However, small language can't support the everything that user wants such as server communication, GUI, or threads. Small, growing programming languages can achieve this goal. The new "words" added by users grow the languages. This added "words" can be used for other users.

We need to ask, "How the language will grow?". We might update the "vocabulary" or "rules". Libraries are the "vocabulary". Getting help from many users can help the language grow fast. There might be two strategies to do so. Many workers work together with one designer who keeps the plan. On the other hand, many workers can build without a specific supervisor, not considering who is building (selling) what. There are pros and cons, but language usually grows with the second pattern. The plan can be changed while working on it.

However, it doesn't mean we do not need the design. What we need to design is how to act and how to design. Users will work hard, be inspired by others, or might sense the required changes.

The choices that are made need to leave the space of the other choice. We need to design these strategies. Being responsible for what we are using is the key. To do so pattern is required.

The pattern is used to identify. The pattern can give insight into how it works, and when it is best to use. The pattern is a design space that can you choose on the fly. Within the pattern, we can grow the languages.

Looking at the Java Programming Language specifically, Java lacks some language characteristics. Java does not have generic types. Operating overloading is not supported, too. It would be the best.

However, what about the same language types that are being used by users? Rational number? Interval? They are used by users and have specific rules. It is not a good idea to support these features or data types as the Java Language primitives.

An ideal way to build programming languages is like a shopping mall. We need to think about the term "Meta – you step back from your place". Ultimately, language design is the pattern. The growing pattern. Therefore, programming language needs to be a small language that encourages users to grow the language as they want. Programmers build new vocabulary, and add it to programs, and the programming languages only need to support it.

Ultimately, a new language is built on the languages. During the procedure of growth of the language, one or more groups test and judge what we have added. The ones in society need help with the task of growing. Need to have a plan to cooperate with others.

Java was a small language. Big language provides the burden to programmers to learn the languages. Small language is too small to support what we users need. As previously mentioned, the essence of language design is not designing "the right thing". User doesn't wait for the right thing, just choose the cheap and fast. Therefore, the right thing is to provide the tools to grow the language.

Overall, programming languages are not that different from the language we are using. Some blocks became the base, and the base formed the new features. To express and combine, rules are defined. Programming languages, viewed as lists of instructions for computers, grow through user contributions. Also, it is important to focus on designing patterns that accommodate expansion. Programming language should be a small framework encouraging user-driven growth, with the evolving pattern being the essence of language design. Therefore, the key point of designing programming language is making the environment to encourage to participate and give the tools to grow the language.