1 概要 と アブストラクト

1.1 概要

新しく設計・開発される情報システムは、機能が益々高度で大規模になる一方で、開発期間を短縮することが要求されている。

本研究では、この問題に対して、アクティブソフトウェア (Active Software) の考えに基づいて、大規模で安全なシステムの新しい設計法を確立すること、その開発環境を構成することを目指している。

三年間の研究の結果、次のように、安全で柔軟な構成をもつソフトウェアの新しい設計方法と、それに適したプログラミング環境およびコンピュータアーキテクチャの総合的な基盤のひとつを実現した。

(1) 仕様や環境の変化に対応して、常に安全に稼動するソフトウェアに適したアルゴリズムの開発と設計法

仕様に添って安全に実行することを保証する事前条件・事後条件に加えて、予想していない状況に対応する事前チェック・事後チェックによる記述を導入し、それに合わせたアルゴリズムの設計法を研究した。

条件やチェックの検出には、起動条件を備えた「能動関数」によって表現し構成している。

(2) アクティブソフトウェアの設計解析法とその記述の言語処理系の開発

目的とするソフトウェアの動作を解析した結果を状態遷移図あるいは π 計算式で表現する。それを能動関数によるプログラムの枠組みに変換するツールと、そこから、C および C++ に変換する言語プロセッサを連結して、ソフトウェアの実装の支援をする流れを実現した。

これは、イベント駆動型の形式になっている。これにより、ソフトウェアの仕様の変更に伴って、プログラムの構成を変更することを行い易くする基盤を与えている。また、関数の呼び出し関係のリストを出すこと、実行時の呼び出し状況を追跡できるようにすることで、プログラムの構成と実行をわかり易くしている。

(3) アクティブソフトウェアを実行する可変構造ハードウェア環境

アクティブソフトウェアでは、能動関数の起動条件となるイベントの検出方法が実行の効率に大きく影響する。そのため、プログラムに応じて、イベントの検出や例外条件の検出を常に監視する再構成可能なイベント駆動型の新しいコンピュータアーキテクチャの構成や、配線を柔軟に変更できるプロセッサ配列の構成について検討を進め、設計した回路の評価を行った。

三年間の、これらの成果を踏まえて、今後、更なる研究を進め、柔軟で安全に走行するソフトウェアの構成、新しいプログラミング環境とそれに適したコンピュータアーキテクチャの統合的な開発を目指す。

1.2 アブストラクト

New information system becomes higher in function and larger in size, but it requires shorter implementation time.

For these trends, we aim to establish a new design method of large and safe software system and to develop appropriate environment for it.

Through the research in 3 years, we gained one basis as follows.

(1) Design algorithms suitable to safe and adaptive software for the change of requirements

We introduced not only per-condition and post-condition to assure safe operation satisfying the specification, but also pre-check and post-check to provide for unpredictable states.

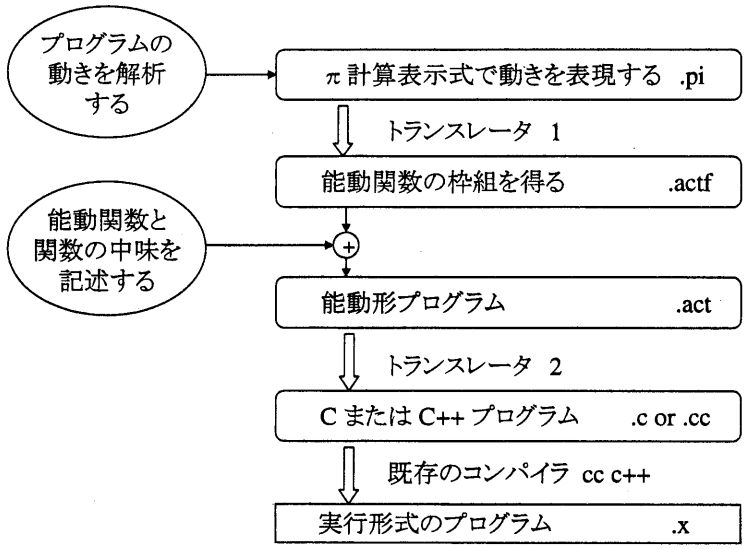

図 1: アクティブソフトウェアの生成の流れ

To detect the condition or check, they are formed with active functions attached each activation condition.

(2) Language to design and analysis active software, and its language processor

At first we analyze the behavior of the software and express the result in state transition diagrams or π-expressions. Then we get the framework of the program with active functions by 1st-translator, append the detail contents of each function by hand, and gain the program text in C or C++ by 2nd-translator.

The obtained program has the feature of event driven activation and is easy to change. The 2nd-translator is able to show the relation of calling and called functions, and also to make the program traceable the real calls at running time.

(3) Re-configurable hardware architecture for active software

On active software the performance is affected by the method to detect events as activation condition of each active function. We proposed new architecture of event driven computer with re-configurable part and array of processors with changeable wire connection between processors, and evaluated the designed results of them.

With these results through 3 years, we have some plans to continue the research about flexible and safe software, new programming environment, and suitable computer architecture for them.