論文内容の要旨
論文題目：
A Type Thoery for Optimizing Compiler (最適化コンパイラのための型理論)

氏名　：　松野　裕

本論文は、コンパイラ最適化におけるプログラム解析およびコンパイラ最適化の正しさの検証ための新たな型理論を、変数間の依存関係を型として抽象化することにより構築する。特に従来課題となっていたループなどにおける再帰的計算を再帰型により抽象化した型システムを静的単一代入方式中間言語において定義し、健全性定理として最適化前後において返り値の型が適切な順序関係を満たすならばプログラムの意味が保存されることの厳密な証明を与える。また提案する型集合のサブセットを導出する型システムを通常の中間言語上で定義し、そのサブセットが静的単一代入方式中間言語より得られる静的情報を包摂することの厳密な証明を与える。

本研究の背景の一つは静的単一代入形式 (Static Single Assignment form (SSA 形式)) を中心としたコンパイラ最適化の発展である。コンパイラ最適化の目的は高速化であり、そのためにあらゆる高度な技術、テクニックが提案・実装されてきた。しかしながら、そのためのプログラム解析および実装の困難さは、最適化が高度化するにつれ深刻化した。より容易に、効率的に最適化を実装を行なえるための形式化は、実装する現場において強く望まれていたと考える。SSA 形式とは変数への代入文をすべて一意にする中間言語の一種であり、Cytron ら IBM の研究者たちにより 1980 年代後半に提案された。コンパイラ最適化の基本は use-def 解析であり、最適化の適用の多くの場面において用いられる。中間言語を SSA 形式に変換することにより、各プログラムポイントで use される変数の代入文は一意に定まり、dead code elimination, common subexpression elimination など、多くの最適化の適用が簡略化された。このことから SSA 形式はコンパイラ最適化における標準的な中間言語として認識され現在に至っている。さらに、use-def 解析はコンパイラ最適化だけでなくアセンブラ言語など低レベル言語におけるあらゆるプログラム解析の基本であ

り、そのため SSA 形式はコンパイラ最適化の適用のみではなく、その正しさの検証など、ソフトウエアシステム検証などにも適用され始めている。コンパイラ最適化の適用を主として、SSA 形式は、近年アセンブラ言語など低レベル言語におけるプログラム解析において重要性が増しつつある。

本研究のもう一つの背景は、型理論を中心とした基礎理論に基づく低レベル言語におけるソフトウエア検証研究の発展である。インターネットの普及によりネットワーク上をバイナリコードが飛び交い、不特定多数のコンピュータ上で実行されうる現在、バイナリコードなどの低レベル言語における安全性検証が必須の課題となっている。しかしながら Bytecode Verifier などの先進的な安全性検証機構を備える Java などにおいても、Java のアプレットの欠陥をついたハッキングによる被害などが発生している。そのため近年より精密な検証を目的とした型理論などの厳密な基礎理論に依拠した低レベル言語におけるソフトウエア検証研究が注目を集めている。代表的な例として Necula による Proof Carrying Code （PCC) や Morrisett らによる Typed Assembly Language(TAL) などがある。PCC や TAL は主に関数型言語研究における様々な成果を応用することにより、メモリ保全、プログラムの異常停止の実行前の検出などの低レベルコードの安全性の検証を試みた。このような方向性は Hoare が型理論や数理論理学に基づく検証つきコンパイラ（Verifying Compiler）の構築を 21 世紀の計算機科学におけるグランドチャレンジとして提唱するなど、計算機科学において大きな潮流となりつつある。

しかしながら、コンパイラ最適化研究においては SSA 形式の登場などを契機として、最適化をより効率よく容易に適用可能にするため、さらには低レベル言語における安全性・正しさの検証のための見通しのよい理論的形式化に関する研究が活発に行なわれているが、残念ながらコントロールフローグラフ上における古典的な形式化の域をこえる従来研究はいまだ存在しない。一方型理論などプログラミング基礎理論に基づく低レベル言語解析は、関数型言語分野における成果を応用することにより行なわれているが、関数型言語と低レベル言語は本質的に異なるため、その試みは容易ではない。

我々は、上記 2 つの研究背景における課題を解決するためには、**従来低レベル言語レベルにおける解析を包摂し、かつ関数型言語における成果に表面的に依拠しない、低レベル言語のための新たな形式的理論体系の構築**が必要であると考える。その構築のため本研究は具体的な対象としてコンパイラ最適化の正しさを検証するための理論体系の構築を目指す。その手段として、我々は低レベル言語の解析において最も本質的なものを抽象化し、それらを導出する形式的な体系を構築することを試みる。形式的体系として型理論を用いる。我々のアプローチは型理論に基づき、プログラムの様々な概念を抽象化することにより新たなプログラム解析理論の構築を目指す近年の研究動向に沿うものであるが、コンパイラ最適化などのプログラム変換に対して本格的に型を導入する研究は少ない。抽象化すべき低レベル言語の本質は、前述のように、変数間の use-def 関係などのプログラムに現れる様々なオブジェクトの依存関係である洞察する。

本研究は上記目的・洞察に基づき、コンパイラ最適化における最も基本的な概念であるデータ依存関係（use-def chain）を変数の代入型として抽象化することにより型システムを SSA 形式中間言語上において構築し、健全性定理として、もし return value の型が最適化前後において適切な順

序関係をみたすならば、プログラムの意味が保存されることの厳密な証明を与えた。さらに、最適化前後の型の変化により、様々な最適化の形式的な定義を与えた。

本論文の貢献は以下の通りである。型システムの構築に当たって、我々は従来研究において低レベル言語レベル言語において形式化困難であった以下の 2 点：

- 破壊的代入（ある変数に対して複数のプログラムポイントにおいて代入文が存在しうること）
- ループにおける再帰的計算

に対して、

- SSA 形式を対象言語とし、
- 再帰的に値が計算される変数に対して、再帰型を与える型システムを構築

することにより理論的形式化を行なうことを試みた。変数の値の計算過程を抽象化した代入型は以下の BNF で定義される。$(\tau_1, \tau_2)_{aop}$ は各算術命令に対応し、$\{l_1 : \tau_1, l_2 : \tau_2\}$ は SSA 形式における ϕ 関数に対応するものである。さらに、$\{l_1 : \tau_1, l_2 : \tau_2\}$ に対して再帰型を導入する。

$$\tau ::= \alpha \mid int(i) \mid (\tau_1, \tau_2)_{aop} \mid \{l_1 : \tau_1, l_2 : \tau_2\} \mid \mu\alpha.\{l_1 : \tau_1, l_2 : \tau_2\}.$$

上記の 2 点は低レベル言語の最も基本的な性質であるが、従来研究においては明確な形式化がなされてこなかったものである。SSA 形式に対して型システムなど形式的な体系を導入する研究は我々が知る限り存在せず、また、再帰的計算に対して明示的に再帰型などの抽象化に成功した既存研究は、特に低レベル言語レベルにおいていまだ存在しない。さらに我々は SSA 形式自体に対する理論的形式化を試みた。提案する型の内、SSA 形式における ϕ 関数に対応する型より定義されるサブセットを導出する型システムを通常の中間言語において定義し、その型情報が SSA 形式が持つ静的情報と等価であることを証明した。

本研究はアセンブラレベルの低レベル言語の基礎的・本質的な性質に焦点を当て形式化を行った。本論文における成果は、アセンブラ言語などの低レベル言語に対する理論的形式化のための、本質的な第一歩になると考える。