報告番号	※甲　　第　　号

主　論　文　の　要　旨

論文題目　　関数型プログラミング言語における通信記述の型付け

氏　　名　　今井　敬吾

論　文　内　容　の　要　旨

本論文は，信頼性の高い分散システムを構築するための通信記述の基盤を与えることを目的として，プロセス計算の1つであるπ計算に基づくプログラミング技法を提案する．具体的には，関数型プログラミング言語Haskellにおいて，非同期π計算に基づく通信コンビネータを実装する手法と，π計算のセッション型システムをHaskellの型システム上で実現する手法を示す．さらに，セッション型システムの理論的な基礎付けとして，主部簡約を定式化し証明する．

今日において，分散システムは極めて一般的である．代表的なものは，インターネットとそれを活用したアプリケーションであり，様々な産業に深く根ざし現在でもその利用は拡大し続けている．今日における自動車は複数のECUが連携して動作する，高い信頼性が要求される分散システムの一つである．分散システムは，その複雑さにも関わらず様々なレベルにおいて産業と生活の基盤を成しているため，信頼性を備えているだけでなく，効率よく構築されることが求められている．

分散システムでは，複数のノードそれぞれにおいて計算が自律的に進行する．個別のノードが役割を果たし分散システムが全体として正しく動作するためには，各ノードの内部動作が正しく進行するだけでなく，必要なメッセージが適切なタイミングで交換される必要がある．しかしながら，分散システムの連携動作は並行性をもち非決定的であるため，通信のみに着目した場合でもシステムが取り得る状態の数は膨大な数に上る．

分散システムの通信動作を網羅的に解析するためには，個別の通信記述が全体として整合していることを確認しなければならない．ここで通信記述とは，ノードの振る舞いに関する記述のうち，通信に関する部分である．通信記述の整合性を確認するためには，陽に，ないし暗黙に定められた通信手順との適合性の検査が必要となる．インターネットの中心的存在であるHTTPや，SMTPなどは代表的な通信

手順である．分散システムの開発においては，厳密な計算モデルに基づきソフトウェアを構成し，通信記述の整合性を解析できることが望ましい．

プロセス計算は，プロセスと呼ばれる計算主体の間で発生する通信に着目した形式的な計算体系である．これらの体系においては，計算は各々のプロセスが保持する通信路を媒体とした同期的な通信により進行する．MilnerによるCCS，HoareによるCSP，Milner, Parrow, Walkerによるπ計算などが知られている．形式的には，名前が通信路の端点として扱われ，計算は各々のプロセスが保持する名前を介した同期的な通信により進行する．

π計算は通信路を介して名前それ自体を送受する機構を備えたプロセス計算である．これにより，プロセス間のリンク構造の動的な変化を表現できる．π計算では，名前の送受や生成を行う並行プロセスを柔軟に記述でき，型理論やプロセスの等価性に関する多くの解析技法が開発されている．さらに，名前によりポインタや参照といった計算資源をも表現できる．このため，π計算は様々な抽象度において並行分散ソフトウェアの形式的な基礎として多く用いられる．

Haskellは強力な抽象化能力をもつ関数型プログラミング言語の一つである．Haskellは強く型付けされた言語であり，厳密な型検査の能力をもつ．このため，Haskellで記述されたソフトウェアは基本的に信頼性が高いとされている．さらに抽象化の技法として，関数型プログラミング言語ではコンビネータや埋め込み言語などと呼ばれる技法が知られている．例えば構文解析やリレーショナルデータベース等の特定分野における語彙を，言語処理系を拡張することなく実現できるため，実装者と利用者の双方に利点がある．

本論文で提案する枠組みにより，静的型付けの信頼性のもとで通信記述の整合性を確保でき，通信について信頼性をもつ分散システムを構築できる．さらに，既存の柔軟かつ信頼性の高い静的検査をもつHaskellを利用するため，利便性と通信の信頼性を両立できる．

まず，この構想の基礎として非同期局所化π計算(ALπ)を基礎としたネットワークプログラミングの枠組みを提案する．この枠組みの特徴は，Haskellの型システムにより，ALπの通信路が型付けされ，内部動作と通信の両面において実行時エラーが発生せず信頼性が高いことである．さらに，フレームワークの実装においてALπの動作意味を直接的に模倣しており，π計算における既存の技法を適用して振る舞いを解析できる．通信記述においては，π計算がもつ表現能力をそのまま利用できることが望ましいが，計算資源の多くは計算機に局所化されており，名前とアドレスやポートといった計算資源を一対一に対応づけられない．ALπは，通信プリミティブを非同期に制限し，さらに通信路の入力能力の移動を局所的に制限した体系である．名前の局所性により，通信路と計算資源の対応が明白である利点がある．このため，通信記述の基本的な道具として，π計算の名前渡しの能力を使っ

た記述ができる．名前の局所性をHaskellの型システムで表現するために，π計算のサブタイプ関係を表現する型クラスを導入する．

この枠組みにおいては，ALπのプロセスはHaskellのモナドを表現したコンビネータとして提供される．Haskellではモナドはプログラミングの基礎的な道具として極めて頻繁に用いられるため，プロセス計算の構文をそのままプログラムコードに記述するよりも利用しやすいためである．この枠組みの有用性を示すため，インスタントメッセンジャーの例を与える．

一方，通信記述においては，単一の通信路において異種のメッセージを送受信する通信手順に従わなければならない場合も多い．ALπが基礎におく型システムは通信路について同種のメッセージしか扱えず制限が強いため，そのような通信手順に関する静的な解析ができない．

π計算のセッション型システムは，異種のメッセージの系列（セッション）の通信を表現できる型システムであり，そのような通信手順を伴う通信記述の信頼性を静的解析により保証できる．セッション型は通信路に割り当てられた通信手順を表現し，通信手順の違反は型エラーとして検出できる．しかしながら，通常のプログラミング言語の型システムでは，セッション型を直接に表現できない．まず，ほとんどの型システムでは，識別子にただ一つの型を割り当てるため，通信路の使用順序を型で表現できない．さらに，線形型のような型機能が無いため，スレッド間で通信が干渉しないことも保証できない．

このような型をHaskellの型にエンコードするために，様々な挑戦がなされてきた．SackmanとEisenbachによる実装は，豊富な機能を持つものの，セッション型が推論されず，型をソースコードに陽にする必要があり，通信の記述が煩雑になりがちだった．近年のPucellaとTovによる実装は，型推論によりセッション型が推論されるものの，複数の通信路を扱うために本来不要な辻褄合わせの操作をしなければならず，多くの通信路を扱う場合にやや問題があった．

そこで本論文では，従来手法の問題を解決し，型推論を伴うセッション型システムをHaskellの型システム上に実現する手法を提案する．PucellaとTovによる既存の実装[?]では，型推論によりセッション型が推論される．しかしながら複数の通信路を扱うために通信路のスタックを操作しなければならず，多くの通信路を扱う場合にやや問題があった．我々の実装では，de Bruijnレベルによるエンコーディングを用いて，複数の通信路を用いた場合においてもスタック操作などが必要ない，ほぼ全自動のセッション型推論を実現する．枠組みの有用性を示すため，セッション型で型付けされた複数の通信路を用いたSMTPクライアントの実装を示す．

さらに，我々のセッション型の実装の理論的基盤を堅固にするために，本論文ではセッション型システムの主部簡約について異なる定式化を行い，その型安全性を確かめる．本田らにより提案されたオリジナルのセッション型システムは，型シス

テムの基礎的な性質である主部簡約が成立しない場合があることが，吉田らにより指摘されている．我々は本田らのセッション型システムに対する修正を与え，主部簡約を定式化し，これを用いて型安全性が成立することを示す．鍵となるアイデアは，型がつかないプロセスに簡約される通信パターンにおいても，依然として安全性が保たれていることを示すために，型付けを拡張することである．具体的には，プロセスの部分項において型安全が成立しない場合においても，その部分には到達しないことを主部簡約として定式化し，その証明を用いて，型安全性を示した．

我々の技法により，Haskellの従来の静的型付けに加えて，セッション型システムによる通信手順の整合性検査がHaskellコンパイラにより提供される．一連の結論として，分散システムを，高い信頼性で，かつ効率よく記述できるようになる．