Theoretical Computer Science 卒論用

１．Introduction 導入

リバーシブルフローチャートは、従来のフローチャートが、幅広く使われているプログラミング言語のモデルであるように、可逆プログラミング言語のモデルとして意図されている。

フローチャートは、制御フローと、命令型のプログラムの構造を表している。但し、常に標準計算の基本理論を可逆言語に直接引き継げるとは限らない。

その結果、古典的なモデルを研究することで、可逆プログラミング言語を直観的に理解できるとは限らない。

例えば、Axelsen や Gluck から、どの関数が計算できるかを考慮する時、可逆プログラミング言語はチューリング完全ではない。更に、構成スペースが制限されている時、可逆プログラムは常に終了する。

さらに、標準な言語理論で当たり前だと思っている古典的な結果を、すぐに明らかにできない。すなわち、構造化されたプログラムの定理が可逆的な設定に引き継がれるということ。

リバーシブルフローチャートは従来のフローチャートと表面的に似ているにもかかわらず、それらは重要な違いがある。原子ステップは局所的に可逆な操作に限られていて、ジョインポイントは、明確な直交条件の式を求められている。

どのステップの計算プロセス中でも、情報が失われることがなく、全てのステップが、順方向か逆方向に決定的になる。これらにより、リバーシブルフローチャートが、計算能力と、可逆プログラミングの単純な特性や、理論的で、実用的な従来の形式とは違う研究をする実りの多い意味のある形式を正確に捉えられる。

フローチャートは同じフレームワークで、かなり多様なプログラミング言語の概念を満たす大きな利点がある。リバーシブルフローチャートはかなり広い目的で使われている。コンパイラによって生成されたマシンコードの低レベルな側面と、高いレベルの汎用的なブロック構造化言語や、反復や条件付き文に対応する。

さらに、モデルは、可逆ロジックデバイスや可逆マイクロプロセッサなど、基礎となるコンピュータハードウェアの具体的な実現や編成とは無関係である。

そのグラフィカルな形式は、可逆メモリ管理や、RTMのヒープ操作を説明するのに役立つ。

（可逆言語の部分的な評価、制御フロー、様々な可逆言語も同様に。）

このような言語の本質の共通の形式を蒸留することは、計算スタックの様々なレベルで可逆言語を設定する基礎となり、プログラムインバイダーや様々な可逆アーキテクチャのコンパイラーなどの可逆プログラミング用のツールになる。

彼らの独特なプログラミング方法論は、プログラムのモジュール性や、新しいスタイルに光を当てるかもしれません。

この論文の二次的な目的は、可逆計算の論理的および実用的な性質に関する将来の取り組みを奨励することである。

可逆性の別のアプローチは、「広がりの下にごみを隠す」で、継続的に情報をつくる完全な制御をランタイムシステムに提供し、ユーザーの操作や、制御なしに可逆性を保証することである。

私たちはここではこのようなアプローチはとっていない。しかし、ごみデータの管理を言語レベルで明確に処理する問題とみなしている。それらはよく可逆的な概念を利用するが、プログラムの逆転や、双方向変換といった関連分野は、より従来の言語のコンテキストで適用される。一般的に、プログラムの逆転は、不可逆プログラムと双方向変換の操作は、与えられた情報を返すことができる不可逆プログラムに関する予測がある。これは、この論文の範囲外になる。

従来のフローチャートと同じように、構造は単調な可逆スパゲティコードに必要である。ま た構造は可逆フローチャートの中で存在することができる。 限定的な構造化されていない制御は、３つの構造化可逆制御フロー演算子によって行われ た。それは連続、選択、ループである。 私たちは構造化可逆プログラム定理をこの論文で証明するでしょう。すべての可逆フロー チャートは構造化されたものに対応付けることができることを保証する。これは構造化可逆フローチャートが構造化されていないものと同じくらい強力かつ表現力があるということ、さらにｒチューリング完全を意味している。 構造化されていない可逆フローチャートは低レベル可逆機械コードのモデルとして適して いるでしょう。一方、高レベルブロック構造化可逆プログラミング言語の様々なモデルには 適していない。 高レベル言語から低レベル機械コードへの翻訳機のような道具は実用的なコンピューティ ングシステムに必須である。 可逆言語の世界では私たちは掃除を必要とする。それは変換で生成された目的のプログラ ムにゴミを取り込まない。 掃除の存在は、構造化されたものと構造化されていないものの２つの小さい可逆命令言語 で実証されるでしょう。

私たちはリバーシブルプログラミング言語の計算モデルとしてのリバーシブルフローチャ ートを中心に調査する。読者がこの論文でまとめられた明確な特徴を理解するのを助ける ため図 1 は、主なトピックとその関係の概要を示しています。(シェーディングされた領域 は、リバーシブルフローチャートに関連するパーツを表します) 古典的なフローチャートと 比較して、リバーシブルフローチャートは結合点に制限を置き、ステップ演算子は計算の前 方および後方決定論を保障する。結果として、それらはチューリング完全ではなく、r-チュ ーリング完全であり、すべての計算可能な関数を計算することはできませんが、代わりに、 図の左側に示すように、注入可能な計算可能関数の適切なサブセットを意味します。(明らかに、計算可能な関数の外に挿入関数を終了する) これらの関係は図で示され、ここで、チューリングの完全性と r チューリングの完全性-は、 対応するセットをフローチャートの世界と同一視する。 すべての計算可能な関数は、必要な注入性を保証する追加の出力を追加することにより、 単射可能な計算可能な関数に埋め込むことができます (ささいな注入は、元の入力と出力の 両方で構成されるペアを返します)。 余分な、ただし本来は不要な、出力の一部はガベージ と呼ばれます。 フローチャートの変換を、可逆フローチャートへの変換と呼びます。これ は、数学の世界で言及されている関数の注入に対応しています。 この用語は、これが単射 関数ではなく可逆プログラムを生成することを強調しています。

すべての単射計算できる関数は逆関数をもつ。また、すべての可逆プログラムもそうである。 可逆プログラムの逆プログラムはプログラムの反転によって機械的に得ることができる。 2 つの可逆プログラミング言語間のソース to ソースプログラムインバータは図 1 の右側の 「反転」というラベルの付いた円型の矢印で示されています。 リバーシブルフローチャートモデルのプレゼンテーションは、次のように編成されていま す。セクション 2 では、非構造化および構造化された可逆フローチャートの要素について 説明し、反転、境界空間下での終了、反転、リバーシブル フローチャートの r-チューリン グの完全性などの概念について説明します。実行例として、フィボナッチペア関数が使用さ れます。セクション 3 は、構造化された可逆プログラム定理を証明する。この証明は、まっ たく同じ機能を持つ任意のリバーシブルフローチャートから構造化された可逆フローチャ ートを生成する、クリーンな変換方式を使用します。セクション 4 では、フローチャート 形式主義に基づく具体的な言語として完全な構文と構造的な操作セマンティクスを持つ 2 つの小さな可逆的命令型プログラミング言語を導入します。セクション 5 では、2 つのプロ グラムインバータと構造化言語から非構造化言語への翻訳機を定義します。第 6 節では、 ダイクストラの順列からコードへのエンコーダーの反転、RTM をリバーシブルフローチャ ートにクリーンに変換するなど、これらのツールのアプリケーションを紹介し、2 つの言語 の完全な機能を直接示します。最後に、セクション 7 は関連する作業について議論し、セク

ション 8 は論文を締めくくります。 最終段落 この論文は会議への貢献を大幅に拡張、および改訂したものです。 フローチャートの意味論はより詳細化されており、構造化可逆プログラム定理の証明はよ り正確であり、その構造は改善されており、構造の完全な具体例が示されています。 2つのサンプル言語の正式な意味論が提供され、それらの設計のガイドラインが紹介され ています。 構造化されていないフローチャートから構造化フローチャートへの明確な翻訳が改善され、 両方のサンプル言語の翻訳が表示されます。 可逆的なチューリングマシーンは洗練されており、構造化されたサンプル言語のプログラ ムによって例示されています。 計算モデルの理論的側面のより詳細な議論は、プレゼンテーションを終えています。 この論文で紹介する可逆フローチャート言語の基本が可逆コンピューティングシステムの さらなる開発を促進することを願っています。 この作業は可逆論理入門とその物理的実現を含む可逆コンピューターシステムの開発にお ける大きな取り組みの 1 つです。