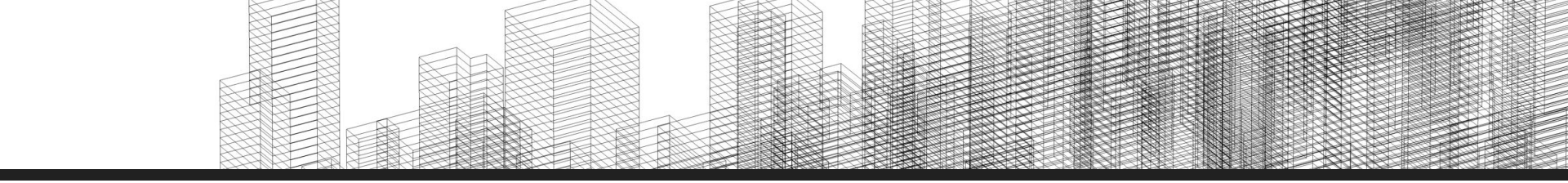


Reversible Space Equals Deterministic Space

Lange, K.J., McKenzie, P. and Tapp, A.

Journal of Computer and System Sciences, Vol.60, No.2, pp.354–367 (2000)

論文紹介　田島嘉人

この時間の目的と概要

この発表は...

- 調査した論文について研究室全体で共有するため
- 研究室での交流を深めるため
- 僕の発表練習のため ...

調査した論文のテーマ: **可逆アルゴリズム**

＊　現在僕が取り組んでいる研究テーマです

他のテーマに取り組んでいる4年生の方や、3年生の方には馴染みが無いかもしれませんが、頑張って分かりやすく説明しますので聞いてくださると嬉しいです

自己紹介

田島嘉人　修士1年

・今年から可逆コンピューティングの研究に取り組んでいます！

LINEのプロフィール画像です

何かのタイミングで連絡したいことがありましたら、こちらにお願いします！

紹介する研究分野について

可逆コンピューティング：可逆な計算を扱う研究

可逆な計算とは...

入力：x → 関数 ：f → 出力：f(x)

入力：f(x) → 関数 ：f® → 出力：x

x から f(x) を求められ、f(x) から x を求められる

例えば... 自然数 n を入力すると n 番目と n+1 番目のフィボナッチ数を返す関数 fib

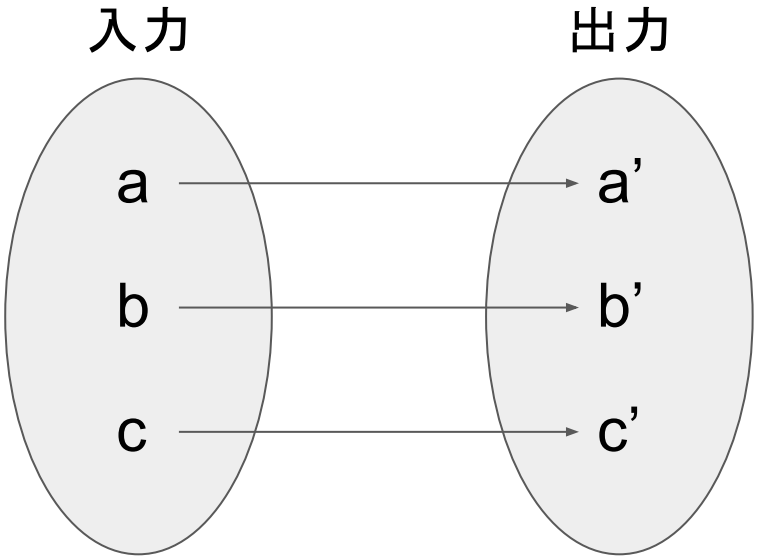
→ fibに5を入れると8と13を返す！

fib®に8と13を入れると5を返す！

紹介する研究分野について

どうしたら計算を可逆にできるの…？

計算を**単射**にする　　単射：異なる入力が同じ出力にならない



単射の例

$f(x)=2x$

単射でない例

$f(x)=|x|$

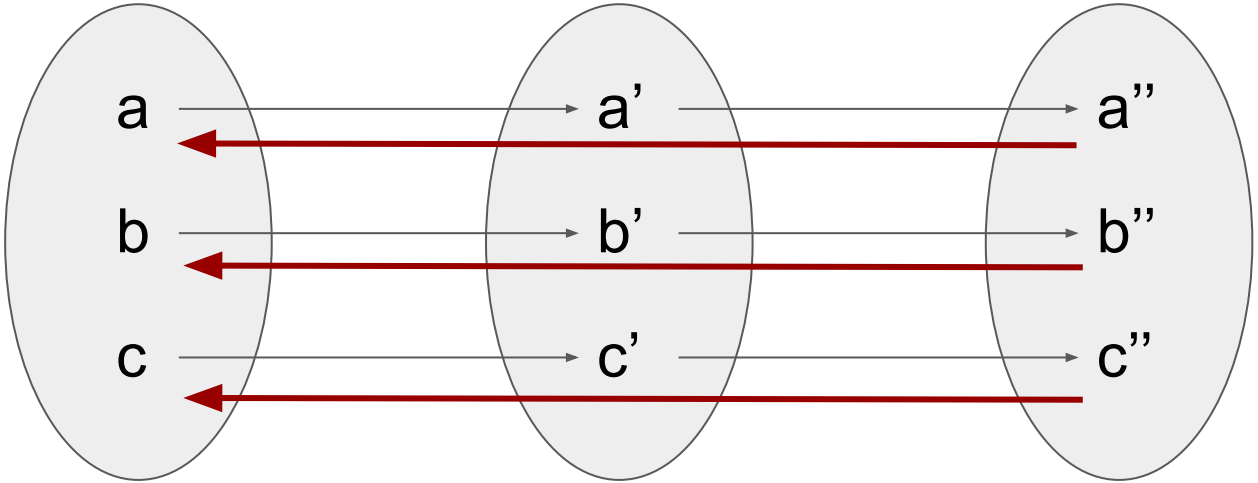
＊ $x=2$ と $x=-2$ のとき、$f(x)=2$

紹介する研究分野について

単射が可逆計算のカラクリ

全ての計算が単射なら、**来た道は一本道**！

つまり、戻る道も一本しかない！



紹介する研究分野について

可逆計算はこんなに便利！

可逆計算の活用例

・マルチプロセッサの投機的実行

・ハードウェアのエラー検出

可逆計算の中・長期的な研究対象

・熱効率のいいコンピュータの開発

・量子コンピュータへの活用

紹介する研究分野についてのまとめ

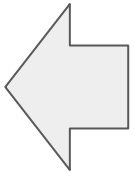
可逆計算とは

計算結果から入力の値を計算できる

全ての計算が単射

活用先が多くとっても便利

便利そうだけど、今まで使ってた線形探索のアルゴリズムとかは単射じゃないから使えないのかなあ…



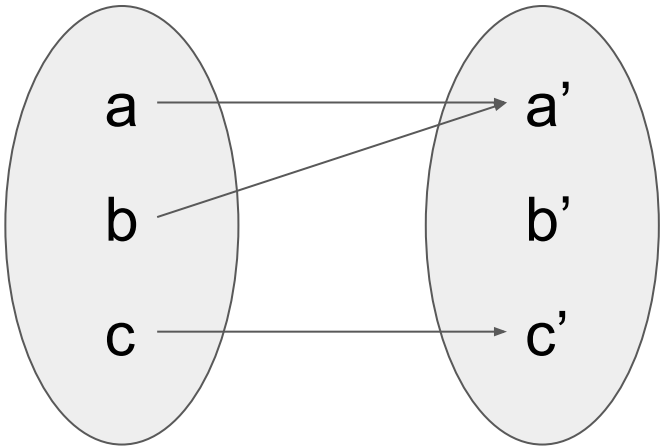
使えます！

非可逆アルゴリズムの可逆シミュレーション

非可逆アルゴリズムには、単射ではない計算が含まれる

—代入・再帰・分岐など...

単射では無い計算



a’に来る道が2本ある...

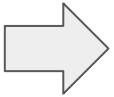
どっちから来たのかわからない...

来た道が分かれば逆計算できる...!?

非可逆アルゴリズムの可逆シミュレーション

計算が辿ってきた道を特定できれば可逆計算できる！

Pebble Game Method



計算の履歴を保存して、辿ってきた道を一意にする

具体的には？

- 履歴を保存するので、シミュレーション前よりも使用する保存領域が増えてしまう...

Reversible Space Equals Deterministic Space

非可逆なアルゴリズムを、履歴を保存せずに可逆シミュレーションできることを示した

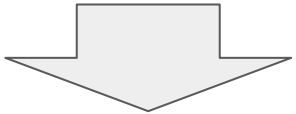
Reversible Space Equals Deterministic Spaceについて

この論文発表以前

・非可逆アルゴリズムは、履歴を保存しないと可逆にはできない...🥺

・履歴の保存量を限りなく削減したとしても S log T が限界だ...🥺

＊入力 n に対して、時間計算量 T(n)、空間計算量 S(n) の非可逆アルゴリズムを可逆シミュレーションした場合



LMT-search(この論文の方法のこと)

履歴の保存量を 0 まで削減可能

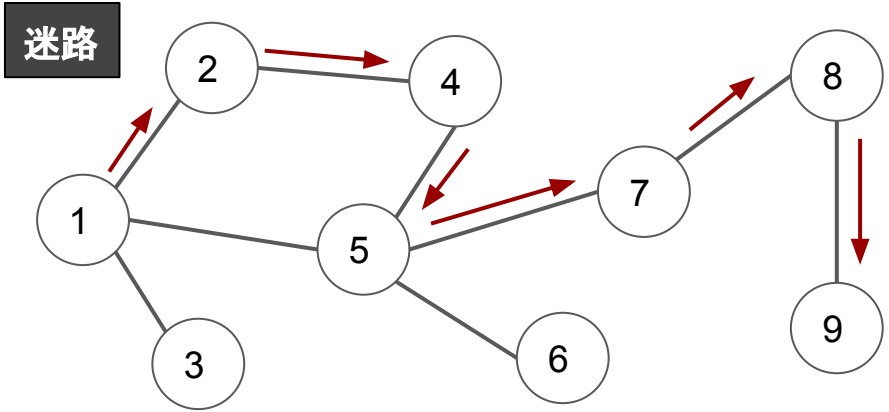
＊正確に表現すると空間計算量S(n)

どうやって削減したのか(アプローチ)

一言で表すと...

左手法を使用した🙋

左手法： 迷路攻略法の基本であり、その内容は「壁に常に左手をつけた状態で前進する事でゴールまでの道を見つける」というもの[引用＊]



絶えず左手を壁に添えることで

1→2→4→5→7→8→9

で迷路クリア！

帰るときは右手を添えれば来た順路の逆順で帰ることができる

⇨ 可逆!?

＊: ニコニコ大百科, https://dic.nicovideo.jp/a/左手法

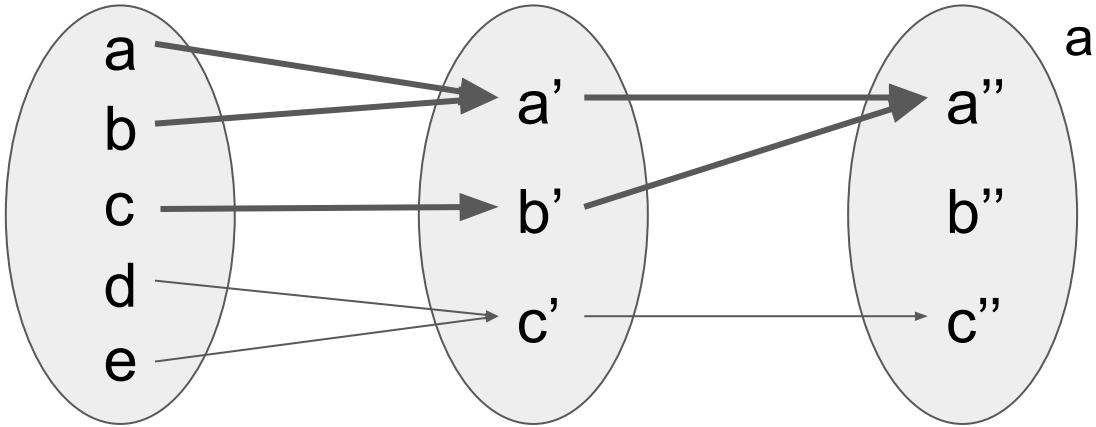
どうやって削減したのか(アプローチ)

左手法

迷路を左手で、道順が1つになるように攻略できた

計算にも応用できる？？

非可逆なアルゴリズムのイメージ



a”につながる矢印を見てみると…

a”を根とする木構造に見える

非可逆なアルゴリズムにも
左手法が使えそう！！

どうやって削減したのか(アプローチ)

左手法が使えそうなのはわかったんだけど、どうすればそれを証明できるんだろう？

計算モデルを使って証明！

計算モデルとは...

・計算を理論的・抽象的に行うことができるもの

・理論の証明に使える

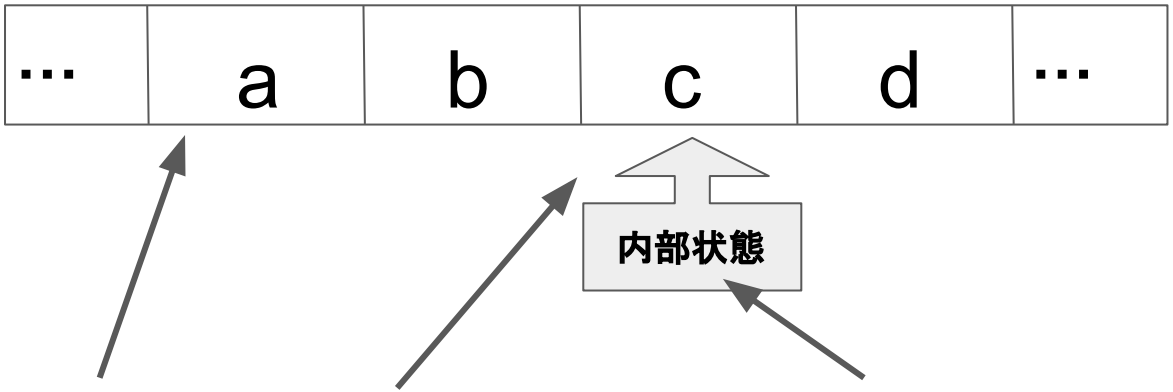
チューリング機械で左手法が使えることを証明

この論文では...

どうやって削減したのか(アプローチ)

チューリング機械とは...

Alan Turing が考案した、どんな計算でも行うことができる思考上の機械



テープ と **ヘッド** と **ヘッド内の状態** で計算する

たったこれだけで、**実行可能な全ての計算が計算できる**

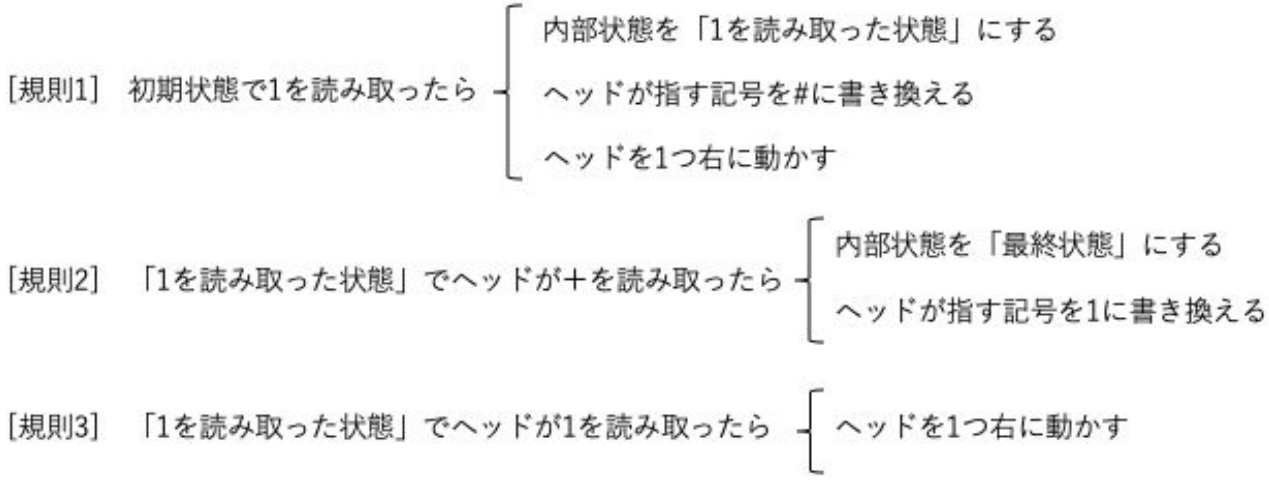
どうやって削減したのか(アプローチ)

チューリング機械はどうやって計算するの？

　めためた簡単な足し算をするチューリング機械を考えてみます

めためた簡単な足し算をするチューリング機械

チューリング機械の動作規則：読み取った情報から何をするか決める

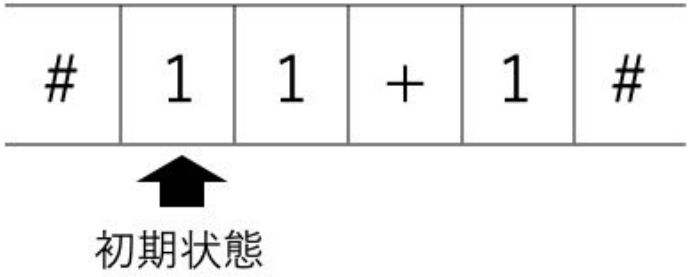


どうやって削減したのか(アプローチ)

めためた簡単な足し算をするチューリング機械

計算の流れ：どうやって計算するのか流れを見てみます

1.



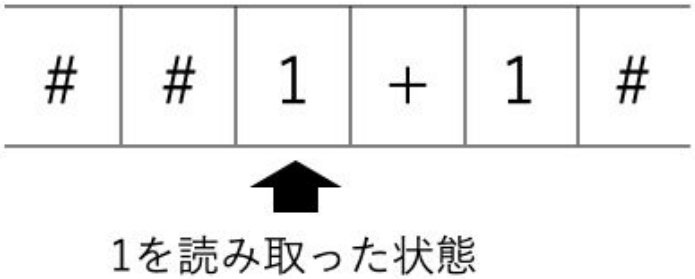
[規則1] 初期状態で1を読み取ったら
- 内部状態を「1を読み取った状態」にする
- ヘッドが指す記号を#に書き換える
- ヘッドを1つ右に動かす

[規則2] 「1を読み取った状態」でヘッドが＋を読み取ったら
- 内部状態を「最終状態」にする
- ヘッドが指す記号を1に書き換える

[規則3] 「1を読み取った状態」でヘッドが1を読み取ったら
- ヘッドを1つ右に動かす

現在 内部状態：初期状態　　ヘッドの位置：1を指している

[規則1]にしたがって



内部状態：1を読み取った状態

ヘッドが指す記号を#に

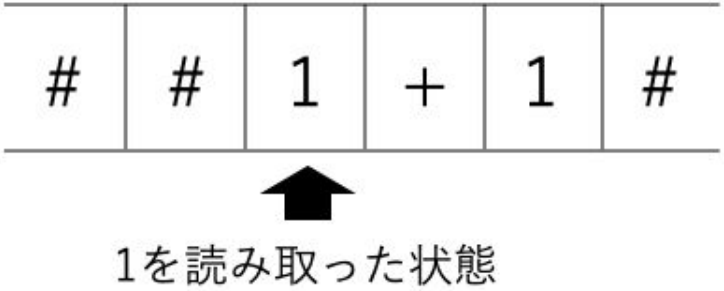
ヘッドを1つ右へ動かす

どうやって削減したのか(アプローチ)

めためた簡単な足し算をするチューリング機械

計算の流れ:どうやって計算するのか流れを見てみます

2.

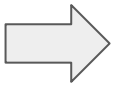


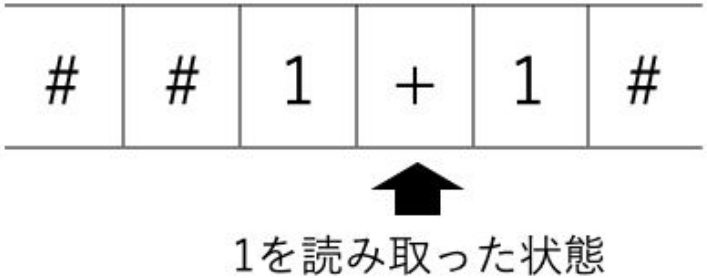
[規則1] 初期状態で1を読み取ったら
- 内部状態を「1を読み取った状態」にする
- ヘッドが指す記号を#に書き換える
- ヘッドを1つ右に動かす

[規則2] 「1を読み取った状態」でヘッドが+を読み取ったら
- 内部状態を「最終状態」にする
- ヘッドが指す記号を1に書き換える

[規則3] 「1を読み取った状態」でヘッドが1を読み取ったら
- ヘッドを1つ右に動かす

現在 内部状態:1を読み取った状態　ヘッドの位置:1を指している





ヘッドを1つ右へ動かす

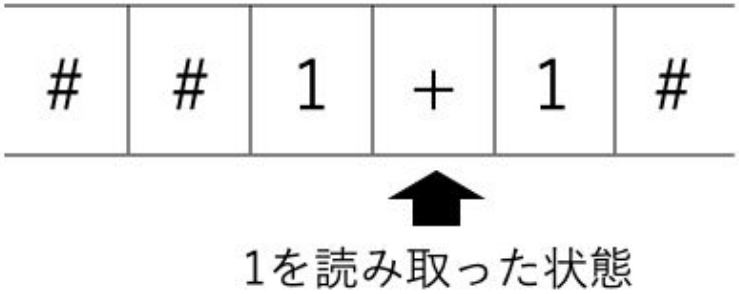
[規則3]にしたがって

どうやって削減したのか(アプローチ)

めためた簡単な足し算をするチューリング機械

計算の流れ:どうやって計算するのか流れを見てみます

3.

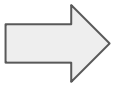


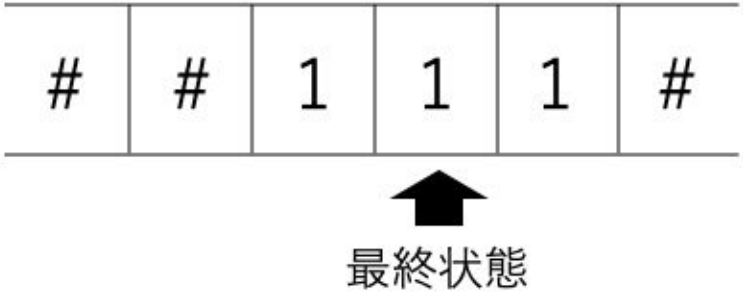
[規則1] 初期状態で1を読み取ったら
- 内部状態を「1を読み取った状態」にする
- ヘッドが指す記号を#に書き換える
- ヘッドを1つ右に動かす

[規則2] 「1を読み取った状態」でヘッドが+を読み取ったら
- 内部状態を「最終状態」にする
- ヘッドが指す記号を1に書き換える

[規則3] 「1を読み取った状態」でヘッドが1を読み取ったら
- ヘッドを1つ右に動かす

現在 内部状態:1を読み取った状態　ヘッドの位置:+を指している





[規則2]にしたがって

内部状態:最終状態

ヘッドが指す記号を1に

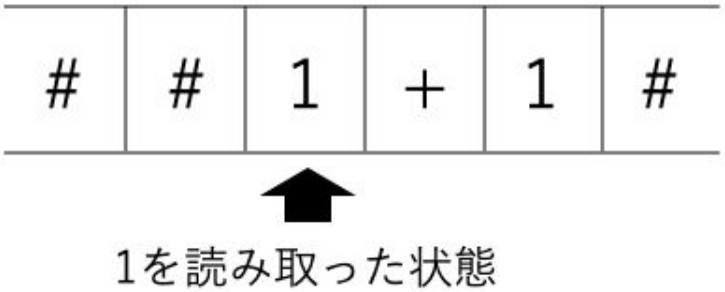
計算終了

どうやって削減したのか(アプローチ)

チューリング機械の様相とは...

テープの記号列・ヘッドの位置・内部状態の組み合わせ

例えば、下の図では...



内部状態：1を読み取った状態

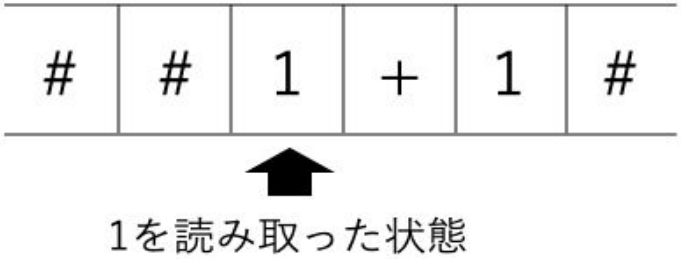
テープの記号列：＃＃１＋１＃

ヘッドの位置：左から3列目

チューリング機械の様相

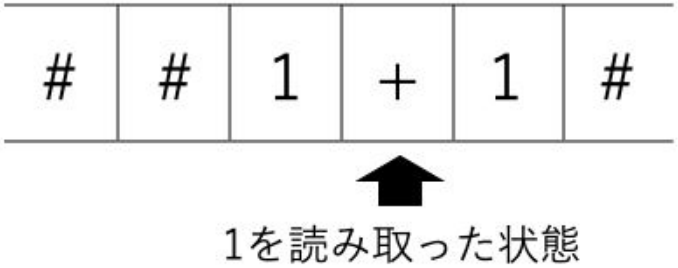
どうやって削減したのか(アプローチ)

様相は計算が1ステップ進む毎に変わる



内部状態：1を読み取った状態
テープの記号列：＃＃１＋１＃
ヘッドの位置：左から3列目

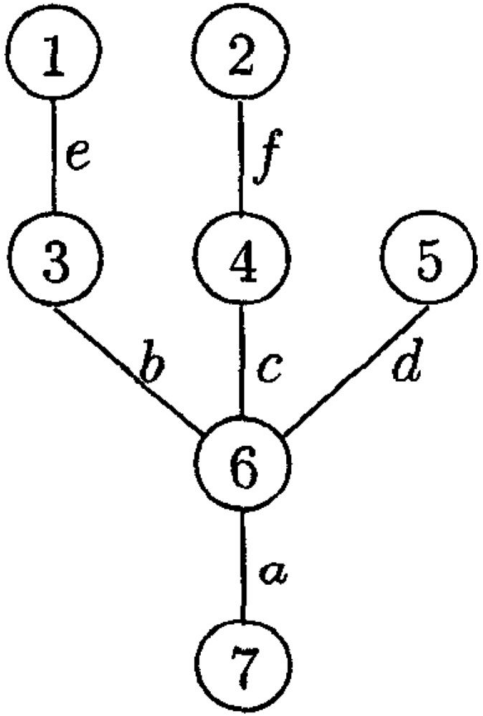
計算が1ステップ進むと…



内部状態：1を読み取った状態
テープの記号列：＃＃１＋１＃
ヘッドの位置：左から4列目

どうやって削減したのか(アプローチ)

一般的なチューリング機械*の様相の移り変わりをグラフにしてみると...



木構造になる！

1, 2, 5 初期様相

7 最終様相

左手法を使えば道順を一意にできる！

(可逆化)

空間計算量 S(n) のチューリング機械の様相遷移グラフに左手法を適用すれば 空間計算量 S(n) で可逆的に計算が終了する！

*決定的チューリング機械のこと

どうやって削減したのか(証明)

チューリング機械の動作規則などを変更せず左手法が適用できることを示す

チューリング機械: $M = (Q, \Sigma, b, \delta, q0, qf)$

形式的定義について詳しくは、
Towards a reversible functional language
計算における可逆性 -可逆チューリング機械と可逆論理回路 -

次の遷移関数を追加

$$\langle \sigma, [q^{\leftarrow} a, qb], s\rangle \to \langle \pi, [q^{\leftarrow} a, qb], t\rangle, -1$$
$$\langle \sigma, [q^{\rightarrow} a, qb], s\rangle \to \langle \pi, [q^{\rightarrow} a, qb], t\rangle, +1$$
$$\langle \sigma, [q^{\leftarrow} a, qb], t\rangle \to \langle \pi, [q^{\leftarrow} a, qb], s\rangle, +1$$
$$\langle \sigma, [q^{\rightarrow} a, qb], t\rangle \to \langle \pi, [q^{\rightarrow} a, qb], s\rangle, -1$$
$$\langle \sigma, [qb, q'^{\leftarrow} b'], s\rangle, b \to \langle \pi, [qb, q'^{\leftarrow} b'], t\rangle, b'$$
$$\langle \sigma, [qb, q'^{\rightarrow} b'], s\rangle, b \to \langle \pi, [qb, q'^{\rightarrow} b'], t\rangle, b'$$
$$\langle \sigma, [qb, q'^{\leftarrow} b'], t\rangle, b' \to \langle \pi, [qb, q'^{\leftarrow} b'], s\rangle, b$$
$$\langle \sigma, [qb, q'^{\rightarrow} b'], t\rangle, b' \to \langle \pi, [qb, q'^{\rightarrow} b'], s\rangle, b$$
$$\langle \pi, [q'^{\rightarrow} b', q'c], s\rangle, b' \to \langle \sigma, [q'^{\rightarrow} b', q'c], s\rangle, b'.$$
$$\langle \pi, [q_i b_i, q'^{\rightarrow} b'], t\rangle, b' \to \langle \sigma, [q_{i-1} b_{i-1}, q'^{\rightarrow} b'], t\rangle, b'.$$

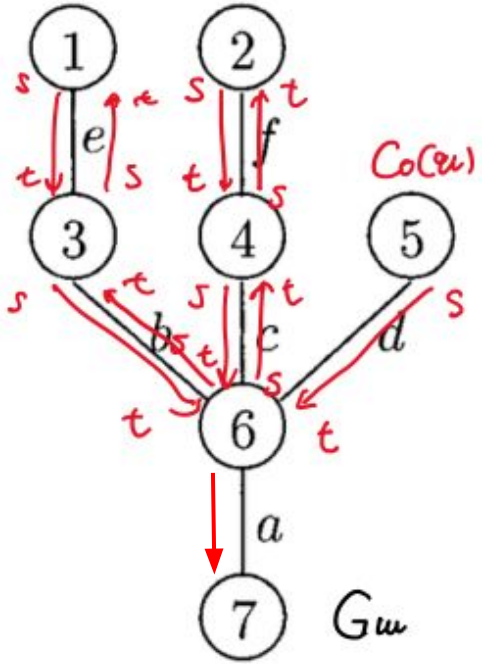
$$\langle \pi, [q_1 b_1, q'^{\rightarrow} b'], t\rangle \to \langle \iota 1.2.1, q'^{\rightarrow}, b'\rangle, +1$$
$$\langle \iota 1.2.1, q'^{\rightarrow}, b'\rangle, c \to \langle \iota 1.2.2, q'^{\rightarrow}, b', c\rangle, c$$
$$\langle \iota 1.2.2, q'^{\rightarrow}, b', c\rangle \to \langle \sigma, [q'^{\rightarrow} b', q'c], s\rangle, -1.$$
$$\langle \sigma, [q'^{\rightarrow} b', q'c], s\rangle \to \langle \iota 1.3.1, q'^{\rightarrow}, b', c\rangle, +1$$
$$\langle \iota 1.3.1, q'^{\rightarrow}, b', c\rangle, c \to \langle \iota 1.3.2, q'^{\rightarrow}, b'\rangle, c$$
$$\langle \iota 1.3.2, q'^{\rightarrow}, b'\rangle \to \langle \sigma, [q_k b_k, q'^{\rightarrow} b'], t\rangle, -1.$$
$$\langle \pi, [q^{\leftarrow} c, qb], t\rangle \to \langle \iota 2.1.1, q, b, c\rangle, +1$$
$$\langle \iota 2.1.1, q, b, c\rangle, c \to \langle \iota 2.1.2, q, b\rangle, c$$
$$\langle \iota 2.1.2, q, b\rangle \to \langle \iota 2.1.3, q, b\rangle, -1$$
$$\langle \iota 2.1.3, q, b\rangle \to \langle \iota 2.1.4, q, b\rangle, -1$$
$$\langle \iota 2.1.4, q, b\rangle, a \to \langle \iota 2.1.5, q, b, a\rangle, a$$
$$\langle \iota 2.1.5, q, b, a\rangle \to \langle \sigma, [q^{\rightarrow} a, qb], t\rangle, +1$$

$$\langle \pi, [q^{\rightarrow} a, qb], t\rangle \to \langle \iota 2.2.1, q, a, b\rangle, -1$$
$$\langle \iota 2.2.1, q, a, b\rangle, a \to \langle \iota 2.2.2, q, b\rangle, a$$
$$\langle \iota 2.2.2, q, b\rangle \to \langle \sigma, [qb, q'^{\leftarrow} b'], s\rangle, +1$$
$$(\text{resp. } \langle \iota 2.2.2, q, b\rangle \to \langle \sigma, [qb, q'^{\rightarrow} b'], s\rangle, +1)$$
$$\langle \pi, [qb, q'^{\leftarrow} b'], s\rangle \to \langle \iota 2.3.3, q, b\rangle, +1$$
$$(\text{resp. } \langle \pi, [qb, q'^{\rightarrow} b'], s\rangle \to \langle \iota 2.3.3, q, b\rangle, +1)$$
$$\langle \iota 2.3.3, q, b\rangle, c \to \langle \iota 2.3.4, q, b, c\rangle, c$$
$$\langle \iota 2.3.4, q, b, c\rangle \to \langle \sigma, [q^{\leftarrow} c, qb], t\rangle, -1$$

どうやって削減したのか(証明)

つまり...??

チューリング機械に様相グラフ上の位置を表す s(ource), t(arget) ができた



⑤ が C0(w)(初期様相) だとすると

1. ⑤にs、⑥にtのパラメータがついている
2. ⑥にs、④にtのパラメータがついている
3. ④にs、②にtのパラメータがついている

⋮

最後. ⑦にtが付き、最終状態を検知して終了

どうやって削減したのかのまとめ

この論文の発表以前

非可逆アルゴリズムを可逆にするには履歴の保存が必要であると考えられていた

この論文

左手法を使い非可逆アルゴリズムを保存せず可逆にできると考えた

証明: チューリング機械を用いてシミュレーション

- チューリング機械にいくつかの遷移関数を追加
- 追加した関数でチューリング機械の様相遷移図を左手法で辿り計算できることを示した

デメリット: 計算量は大きく増える

この発表のまとめ

可逆計算

計算結果から入力の値を計算できる

全ての計算が単射

非可逆アルゴリズムの可逆化

全ての計算を単射にできれば非可逆アルゴリズムを可逆化できる

Pebble Game Method：　時間計算量 **T(n)** 、空間計算量 **Ω(S log T)**

履歴を保存

LMT Search：　時間計算量 **指数関数的** 、空間計算量 **S(n)**

左手法

参考文献

[1]: Lange, K.J., McKenzie, P. and Tapp, A.: Reversible space equals deterministic space,Journal of Computer and System Sciences, Vol.60, No.2,pp.354–367 (2000).

[2]: Yokoyama, T., Axelsen, H.B. and Gl¨uck, R.:Towards a reversible functional language,International Workshop on Reversible Computation,Springer, pp.14–29 (2011).

[3]: ニコニコ大百科 , https://dic.nicovideo.jp/a/左手法

発表は以上です
ありがとうございました！