目次

目次 i

1 素集合データ構造 — 坂口 和彦 1
1.1 素集合データ構造の概要 1
1.2 静的な操作 3
1.2.1 定義 3
1.2.2 補題 7
1.2.3 練習問題 10
1.3 グラフの変更を伴う操作 10
1.3.1 練習問題 15
1.4 まとめ 15

2 形式手法のお仕事 — 平井 洋一 17
2.1 前回までのあらすじ 17
2.2 マイクロカーネルのモデルの検証（某社） 17
2.2.1 プロジェクトの進行 17
2.2.2 プロジェクトの結果 18
2.2.3 退職・職場の消滅 19
2.3 スマートコントラクトの検証 (Ethereum DEV) 21
2.4 今後の戦略 24
2.4.1 仕事と金 24
2.4.2 国 24
2.4.3 ロゴスの光 25

参考文献 27

1 素集合データ構造

坂口 和彦

本章では、**素集合データ構造** (*disjoint-set data structure*) と Union-Find アルゴリズムの Coq での形式化について述べる。素集合データ構造とは、有限集合上の二項関係の同値閉包を計算するためのデータ構造である。また、破壊的変更を用いて、その二項関係に対して新たな情報を効率良く追加できる実装法[*1]の存在も、素集合データ構造の特長である。類似の既存研究としては、素集合データ構造の永続化と Coq による形式化 [2] がある。

素集合データ構造に関する高速化手法として、union by rank と path compression が知られている。本章ではそれらの高速化手法を直接は扱わないが、その形式化の基盤となる定義、補題を含んでいると考えている。

1.1 素集合データ構造の概要

素集合データ構造と Union-Find アルゴリズムは、有限集合 T と、同値関係 $\sim \subseteq T^2$ について、以下の 2 つの操作を提供する。

find $x \in T$ を取り、同値類 $[x] = \{y \in T : x \sim y\}$ の代表元 $x' \in [x]$ を返す。即ち、find は以下を満たす:

$$\forall x \in T. x \sim \mathrm{find}(x)$$
$$\forall x, y \in T. x \sim y \Leftrightarrow \mathrm{find}(x) = \mathrm{find}(y)$$

union $x, y \in T$ を取り、x, y が同じ同値類に含まれるように $\sim$ を更新する。更新後の $\sim'$ は $\{(x, y)\} \cup \sim$ の同値閉包（反射推移対称閉包）である。即ち、以下が成り立つ:

$$\forall a, b \in T. a \sim' b \Leftrightarrow a \sim b \vee (a \sim x \wedge y \sim b) \vee (a \sim y \wedge x \sim b)$$

[*1] ただし、一度追加した情報を効率良く削除することはできない。

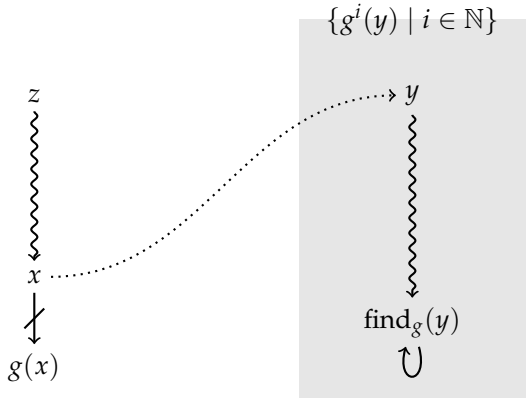

図 1.2: 補題 subst_acycle の図示。波線矢印、打ち消し線付きの矢印、点線矢印はそれぞれ道、削除された辺、新たに追加された辺を表す。ここでは z, x が連結であると仮定し、x から y を指すように変更したグラフ $g' = g[x := y]$ の上での z の弱非巡回性について考える。変更後のグラフでは z から x を経由して y に到達できる。y から $\mathrm{find}_g(y)$ までの道は（y, x の非連結性と補題 subst_wacycle より）元と同じように辿れるので、$\mathrm{find}_{g'}(z) = \mathrm{find}_g(y)$ となる。よって、z は弱非巡回である。

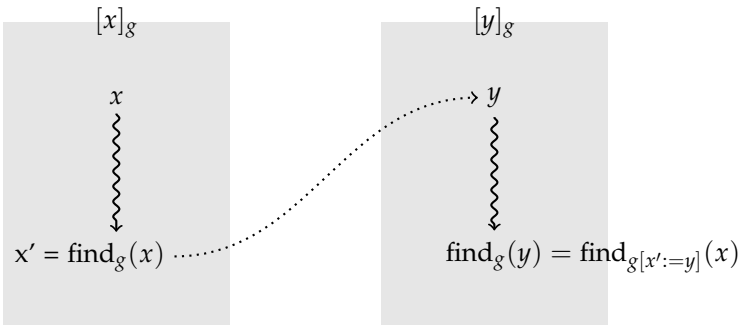

図 1.3: 補題 find_subst_graft の図示。g 上で x, y が異なる同値類（それぞれ $[x]_g, [y]_g$）に属するとき、$x' = \mathrm{find}_g(x)$ から y を指すようにグラフを変更すると x から x', y を経由して $\mathrm{find}_g(y)$ に到達する道が描ける。変更後のグラフでも $\mathrm{find}_g(y)$ は $[y]$ の代表元なので、明らかに $[x]$ の代表元 $\mathrm{find}_{g[x':=y]}(x)$ でもある。

2 形式手法のお仕事

平井 洋一

2.1 前回までのあらすじ

つくばの産総研で働いていたのだが、形式手法エンジニアを雇っている企業があると聞いてドイツに来てしまった。ドイツで働いていると週にほんとうに 40 時間かっきりしか働けないので、のびのびと暮らしていた。

2.2 マイクロカーネルのモデルの検証（某社）

産総研の後で某社のドレスデンオフィスでしていたことは、企業秘密なのでどこにも書けなかったのだが、FM2016 に論文 [1] が通ったので、そこに載っている程度のことは書けるようになった。会社はとある目的のためにマイクロカーネルを開発していた。私のいたチームでは 8 人の定理証明の PhD たちが定理を証明しているはずであった。しかし、実装について定理を証明しようとすると、製品が売り出されるまでに証明が終わらないので、実装ではないもの（モデル）について証明を書いていた。実装とモデルとはテストをして比較するのである。

2.2.1 プロジェクトの進行

最初の数ヶ月はマイクロカーネルの仕様を明らかにするために英語の説明に擬似コードを付けようとしていたが、もとのマイクロカーネルの作者が仕様を非常に簡潔にする主義 [*1] の人だったので、結局疑似コードは仕様書に取り込んでもらえなかった。この擬似コードの構文を毎日延々と相談している [*2] のがど

[*1] この人は同じことは一度しか書かない。そこで、関連することが書いてあっても肝心の欲しい数字は書いていないページがたくさん出現する。ソースコードも非常に簡潔で、ほとんどの変数は一文字である。

[*2] しかも仮オフィスには空調がなくて暑かった。