進捗ゼミ 7/10

SPIKE によるConfluence Property の形式化 の Rocq への移植を目指して

情報科学研究科2年　佐藤龍之介

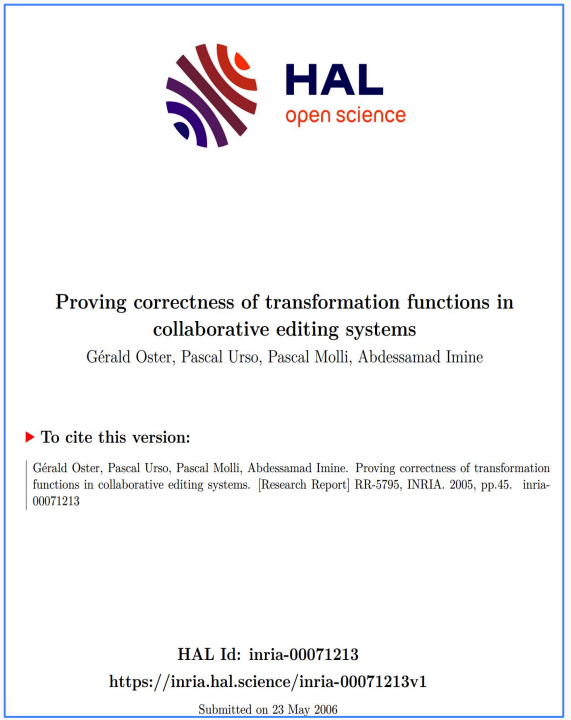
目次

1. 研究背景と目標

・現在の OT のリアルタイムで複数ユーザーによる共同編集に用いられる技術には、Confluence Property の成立が不可欠

・2006年の論文（スライド右）で 自動定理証明器 SPIKE による形式化がすでに行われている

・Rocq でこの形式化を再度行うことで、Rocq を用いた証明の学習になり、さらに Rocq のコードとしての応用の可能性がある



HAL open science

Proving correctness of transformation functions in collaborative editing systems

Gérald Oster, Pascal Urso, Pascal Molli, Abdessamad Imine

▶ To cite this version:

Gérald Oster, Pascal Urso, Pascal Molli, Abdessamad Imine. Proving correctness of transformation functions in collaborative editing systems. [Research Report] RR-5795, INRIA. 2005, pp.45. inria-00071213

HAL Id: inria-00071213
https://inria.hal.science/inria-00071213v1

Submitted on 23 May 2006

引用
：https://inria.hal.science/inria-00071213v1/document

2. 導入

（復習）Confluence Property について

Confluence Property とは、複数ノード間においてデータの順序に関係なくデータの合流が行われる性質

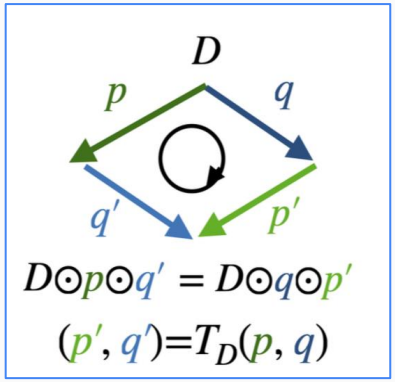


Fig1.TP1 - Property（2ノードにおける Confluence Property）

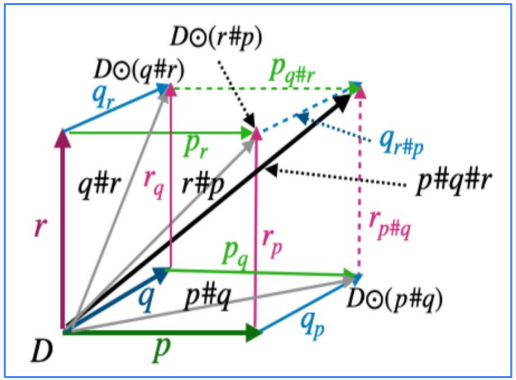


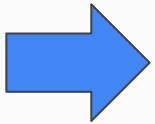
Fig2.TP2 - Property（3ノードにおける Confluence Property）

OT はリアルタイムでの複数ユーザーでの共同編集に用いられる技術

共有されるデータ間の一貫性（ Consistency）を保つ

（アルゴリズムの例） adOPTed, GOTO, SOCT

アルゴリズムの成立には、変換関数（Transformation Function）の定義が必要
もしTP-1,TP-2に関するこの関数が間違っていると、アルゴリズムが共有データの一貫性を保証できない

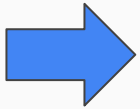


初めに変換関数の正しさを確かめることが重要

変換関数の正しさの証明は難しい！

この正しさを文字のような単純な型のあるもので表すのは困難

さらに、複雑な型を用いて、複雑な操作とともに行うのは不可能



自動定理証明器 SPIKE を用いる

SPIKE の特徴（https://hal.science/hal-02965319/document）

- 自動定理証明器
- 帰納法に基づいた推論
- Sort（データの種類）
- Constructor（データを返す関数）
  - ・Ground Convergence
  - ・Completeness を満たす
- **反例を返す**
- 一階述語論理を用いた Specification の記述（節と等式）

3.OT approach：The Ressel’s model

Divergence Problem

複数ノードにおける最終的なデータの一貫性において問題が発生することがある

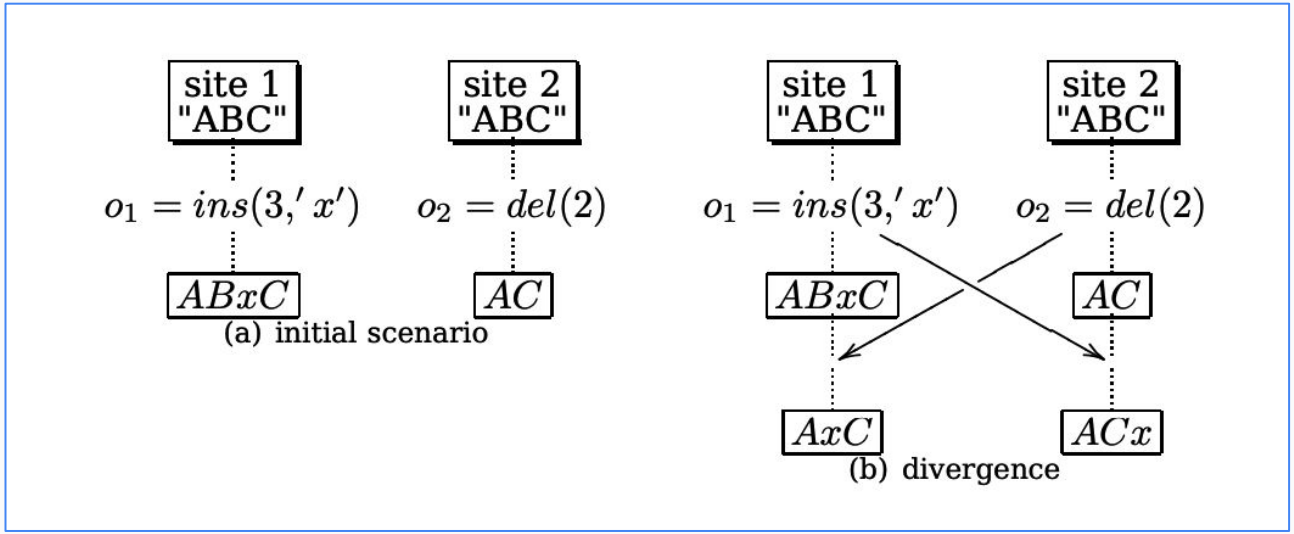
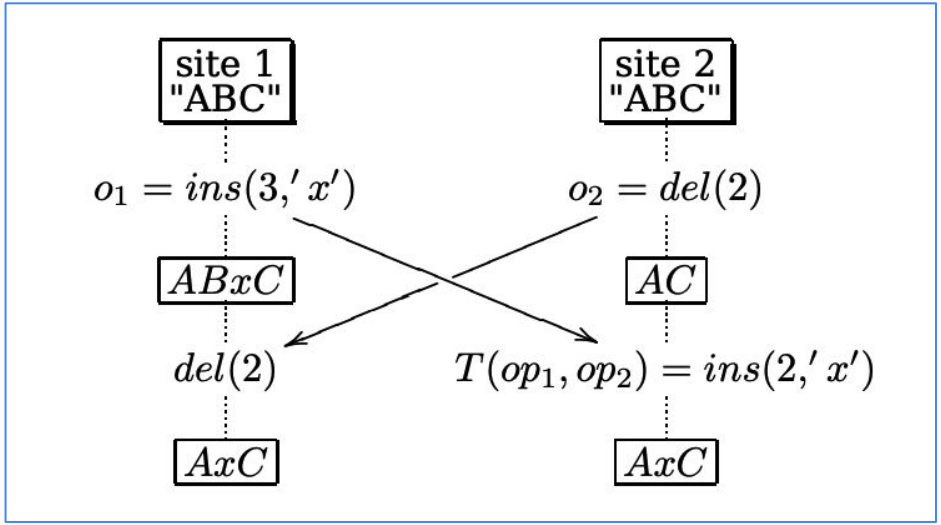


Fig3. Divergence Problem

Divergence Problem の 解決手法

一貫性の問題の解決のために 変換関数Tを導入する



・o_1, o_2 は正しくは op_1, op_2

Fig4. 変換関数Tを導入した図

Divergence Problem の 解決手法

一貫性の問題の解決のために 変換関数Tを導入する

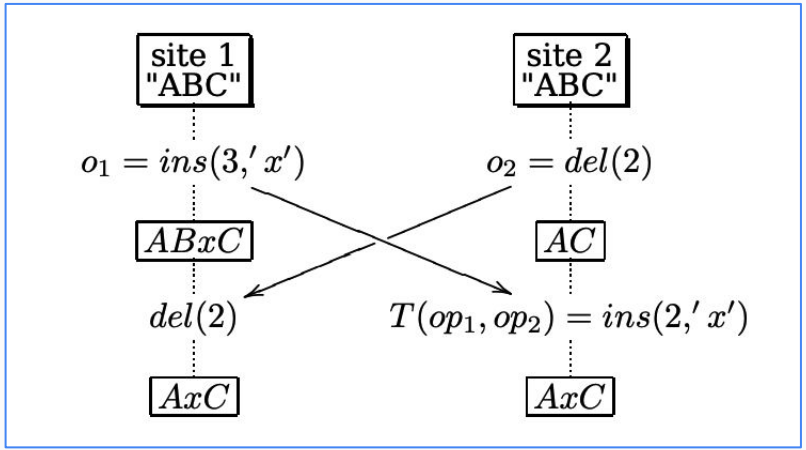


Fig5. Divergence Problem

$T(Ins(p_1,c_1), Del(p_2))$: -
if $(p_1 < p_2)$ return $Ins(p_1,c_1)$
else return $Ins(p_1 - 1, c_1)$

Fig6. Transformation Function T

T の定義

変換関数T は2つの並行な操作をパラメーターとして受け取る

この時のパラメーターをop_1, op_2とし、これらは状態Sで定義される

op_1と等しい$T(op_1, op_2)$は同じ操作だが、状態 $S \odot op_2$ で定義される

Ressel は２つの性質 TP-1,TP-2 を満たす変換関数が並行な操作どうしのいかなる順序においても合流生が成立することを保証した

TP-1 の定義

同じ状態で定義された並行な操作 op_1, op_2 に対して

変換関数Tは以下の時にTP-1を満たす：

$$op_1 \circ T(op_2, op_1) \equiv op_2 \circ T(op_1, op_2)$$

$$\forall S, op_1, op_2 : S \odot op_1 \odot T(op_2, op_1) = S \odot op_2 \odot T(op_1, op_2)$$

TP-2 の定義

同じ状態で定義された並行な操作 op_1, op_2, op_3 に対して

変換関数Tは以下の時にTP-2を満たす：

$$T(op_3, op_1 \circ T(op_2, op_1)) \equiv T(op_3, op_2 \circ T(op_1, op_2))$$

定義より、上と同等な以下のような表現ができる

$$T(T(op_3, op_1), T(op_2, op_1)) \equiv T(T(op_3, op_2), T(op_1, op_2))$$

4.Automated verification of OT functions
- OT関数の自動検証

検証の手順

TP-1,TP-2 に関する変換関数Tの正しさを証明するためのフレームワーク

検証は数学的論理に由来する構文規則に基づいている

1. 検証する変換関数Tの形式的なモデリング
2. TP-1,TP-2 についての関数の正しさを証明する検証
3. SPIKE から得られた結果を解釈するための分析

反例が見つかった時、定理証明器は停止して変換関数Tの修正が必要となる

Ellis et al の変換関数T

Ins(p,c) … インデックスの位置がp番目の所に文字cを挿入する関数

Del(p) … インデックスの位置がp番目の文字を削除する関数

$$
\begin{aligned}
&T(Ins(p_1, c_1, pr_1), Ins(p_2, c_2, pr_2)) :- \\
&\quad \textbf{if } (p_1 < p_2) \textbf{ return } Ins(p_1, c_1, pr_1) \\
&\quad \textbf{else if } (p_1 > p_2) \textbf{ return } Ins(p_1 + 1, c_1, pr_1) \\
&\quad \textbf{else if } (c_1 == c_2) \textbf{ return } Id() \\
&\quad \textbf{else if } (pr_1 > pr_2) \textbf{ return } Ins(p_1 + 1, c_1, pr_1) \\
&\quad \textbf{else return } Ins(p_1, c_1, pr_1)
\end{aligned}
$$

$$
\begin{aligned}
&T(Ins(p_1, c_1, pr_1), Del(p_2, pr_2)) :- \\
&\quad \textbf{if } (p_1 < p_2) \textbf{ return } Ins(p_1, c_1, pr_1) \\
&\quad \textbf{else return } Ins(p_1 - 1, c_1, pr_1)
\end{aligned}
$$

$$
\begin{aligned}
&T(Del(p_1, pr_1), Ins(p_2, c_2, pr_2)) :- \\
&\quad \textbf{if } (p_1 < p_2) \textbf{ return } Del(p_1, pr_1) \\
&\quad \textbf{else return } Del(p_1 + 1, pr_1)
\end{aligned}
$$

$$
\begin{aligned}
&T(Del(p_1, pr_1), Del(p_2, pr_2)) :- \\
&\quad \textbf{if } (p_1 < p_2) \textbf{ return } Del(p_1, pr_1) \\
&\quad \textbf{else if } (p_1 > p_2) \textbf{ return } Del(p_1 - 1, pr_1) \\
&\quad \textbf{else return } Id()
\end{aligned}
$$

Fig7. Ellis's Transformation Function *T*

Ellis et al の変換関数T を SPIKE で形式的に表すと

(RT1) $(p_1 < p_2) = true \Rightarrow T(Ins(p_1, c_1, pr_1), Ins(p_2, c_2, pr_2)) = Ins(p_1, c_1, pr_1);$
(RT2) $(p_1 < p_2) = false, (p_1 > p_2) = true \Rightarrow$
$T(Ins(p_1, c_1, pr_1), Ins(p_2, c_2, pr_2)) = Ins(p_1 + s(0), c_1, pr_1);$
(RT3) $(p_1 < p_2) = false, (p_1 > p_2) = false, c_1 = c_2 \Rightarrow$
$T(Ins(p_1, c_1, pr_1), Ins(p_2, c_2, pr_2)) = Id;$
(RT4) $(p_1 < p_2) = false, (p_1 > p_2) = false, c_1 \neq c_2, (pr_1 > pr_2) = true \Rightarrow$
$T(Ins(p_1, c_1, pr_1), Ins(p_2, c_2, pr_2)) = Ins(p_1 + s(0), c_1, pr_1);$
(RT5) $(p_1 < p_2) = false, (p_1 > p_2) = false, c_1 \neq c_2, (pr_1 > pr_2) = false \Rightarrow$
$T(Ins(p_1, c_1, pr_1), Ins(p_2, c_2, pr_2)) = Ins(p_1, c_1, pr_1);$

(RT6) $(p_1 < p_2) = true \Rightarrow T(Ins(p_1, c_1, pr_1), Del(p_2, pr_2)) = Ins(p_1, c_1, pr_1);$
(RT7) $(p_1 < p_2) = false \Rightarrow T(Ins(p_1, c_1, pr_1), Del(p_2, pr_2)) = Ins(p_1 - s(0), c_1, pr_1);$

(RT8) $(p_1 < p_2) = true \Rightarrow T(Del(p_1, pr_1), Ins(p_2, c_2, pr_2)) = Del(p_1, pr_1);$
(RT9) $(p_1 < p_2) = false \Rightarrow T(Del(p_1, pr_1), Ins(p_2, c_2, pr_2)) = Del(p_1 + s(0), pr_1);$

(RT10) $(p_1 < p_2) = true \Rightarrow T(Del(p_1, pr_1), Del(p_2, pr_2)) = Del(p_1, pr_1);$
(RT11) $(p_1 < p_2) = false, (p_1 > p_2) = true \Rightarrow T(Del(p_1, pr_1), Del(p_2, pr_2)) = Del(p_1 - s(0), pr_1);$
(RT12) $(p_1 < p_2) = false, (p_1 > p_2) = false \Rightarrow T(Del(p_1, pr_1), Del(p_2, pr_2)) = Id;$

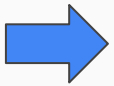
Fig8. Ellis's Transformation Function *T* *expressed in SPIKE formalism*

Step1 ： 形式的なモデリング

定理証明には、定理証明支援と自動定理証明器の2種類がある

- 自動定理証明器を用いると、ユーザの操作無しで事前に構成された戦略を適用して証明を進めることができる
- Specifying （命題の指定）もエラーが起きやすい

しかし、このエラーは変換関数の不確かであることを示すのに役立つ



ユーザは Specification が正しいことの確認と証明器に反例を見つけさせることがメインのタスク

加えて、Specification をある形式から他のものに変換する必要があるが、これを自動で行うツールがある

TP-1 の形式的なモデリング（1）

$$\text{TP-1: } op_1 \circ T(op_2, op_1) \equiv op_2 \circ T(op_1, op_2)$$

TP-1 の等価性を形式的に
以下のように表現できる

$$\forall op_1 \in Op,\ \forall op_j \in Op, \forall st \in State,$$
$$enabled(op_i, st) \wedge enabled(op_i, st) \wedge conc(op_i, op_j) \Rightarrow$$
$$(st \odot op_i) \odot T(op_j, op_i) = (st \odot op_i) \odot T(op_i, op_j)$$

TP-1 の形式的なモデリング（2）

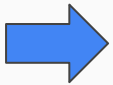
Op は複製されたオブジェクトで定義された操作の集合
State は複製されたオブジェクトに発生する可能性のある状態の集合
$s \odot op$ は状態sに操作opを行なった結果の状態
$enabled(op_i, st)$ は、操作op_iの前提条件が満たされているかを調べる
$conc(op_i, op_j)$は、2つの操作op_i, op_j が並行であるかを決定する
$(pr_1 \neq pr_2)$

$$\forall op_i \in Op,\ \forall op_j \in Op, \forall st \in State,$$
$$enabled(op_i, st) \wedge enabled(op_j, st) \wedge conc(op_i, op_j) \Rightarrow$$
$$(st \odot op_i) \odot T(op_j, op_i) = (st \odot op_j) \odot T(op_i, op_j)$$

Fig9. TP-1 を一回述語論理で表した命題

複製されたオブジェクトの対処法

状態を全て書き出すことは不可能



Situation Calculus（状況計算）...

状態ではなく状態が変化する計算過程 ＝ “**状況**” に注目する手法

状態の複製オブジェクトは、**状態に変化を与える操作** から構築される

Observation function を用いた状況の記述

$car'(n)/Ins(p,c,pr)$ =
  if $(n == p)$ **then return** c
  else if $(n > p)$ **then return** $car(n-1)$
  else return $car(n)$
  endif;

$car'(n)/Del(p,pr)$ =
  if $(n \geq p)$ **then return** $car(n+1)$
  else return $car(n)$
  endif;

Fig10.Definition of Observation function car(pos,st)

(RO1) $n = p \Rightarrow car(n, xSt \odot Ins(p,c,pr)) = c$;
(RO2) $n \neq p, (n > p) = true \Rightarrow car(n, xSt \odot Ins(p,c,pr)) = car(n - s(0), xSt)$;
(RO3) $n \neq p, (n > p) = false \Rightarrow car(n, xSt \odot Ins(p,c,pr)) = car(n, xSt)$;

(RO4) $(n \geq p) = true \Rightarrow car(n, xSt \odot Del(p,pr)) = car(n + s(0), xSt)$;
(RO5) $(n \geq p) = false \Rightarrow car(n, xSt \odot Del(p,pr)) = car(n, xSt)$;

Fig11.Observation function car(pos,st) in SPIKE Formalization

参考文献

- Proving correctness of transformation functions in collaborative editing systems
- Coordination-free Collaborative Replication based on Operational Transformation - Masato Takeichi
- SPIKE, an automatic theorem prover – revisited
- SOME PHILOSOPHICAL PROBLEMS FROM THE STANDPOINT OF ARTIFICIAL INTELLIGENCE（situation calculus に関して）