チュートリアル

『ソフトウェアサイエンスの基本』シリーズ第 4 回

高談闊論：双方向変換の原理と実践

加藤 弘之　胡 振江　日高 宗一郎　松田 一孝

双方向変換とは，ソースデータをターゲットデータに変換した後，ターゲットデータ上の更新をソースデータに反映させることが可能な計算の枠組みのことである．双方向変換の考え方は，古くはデータベース分野におけるビュー更新問題として扱われてきたが，近年は新しいプログラミングモデルと進化的ソフトウエア開発の手法として注目を浴び，プログラミング言語の観点から様々な双方向変換言語が提案されてきた．この解説論文は，会話の形で，プログラミング言語，ソフトウェア工学，データベースの視点から，双方向変換の歴史，基本原理，実践，応用，そして今後の課題について概説する．

A bidirectional transformation consists of pairs of transformations — a forward transformation produces a target view from a source, while a backward transformation puts back modifications on the view to the source — satisfying sensible roundtrip properties. Bidirectional transforms, originating from the view update problem in the database community, are gaining more and more interests from researchers in programming languages as a novel programming model for data synchronization, and researchers in software engineering as a new approach to evolutionary software development. In this article, we would like to outline the history, basic principles, tools, and applications of bidirectional transformations, from different views of programming languages, software engineering, and databases.

1　はじめに

司会: 双方向変換とは，ソースデータをターゲットデータに変換した後，ターゲットデータ上の更新をソースデータに反映させることが可能な計算の枠組みのことです．双方向変換の考え方は，古くはデータベース分野におけるビュー更新問題として扱われてきましたが，近年は新しいプログラミングモデルと進化的ソフトウェア開発の手法として注目を浴び，プログラミング言語の観点から様々な双方向変換言語が提案されていて，まさに，今はやっている，旬の研究ではないでしょうか [9] [29]．双方向変換は，プログラミング言語に関する著名な国際会議 (POPL, ICFP など) とソフトウェア工学に関する著名な国際会議 (ICSE, FSE, ASE など) において近年必ず発表があるばかりでなく，双方向変換に関する国際ワークショップも近年開催されるようになりました．またデータベースにおける著名な国際会議の 1 つである ACM PODS の 2012 年のチュートリアルの 1 つは双方向変換に関するものでした．

双方向変換の魅力の 1 つは身近なところでの応用が色々ありすぐに使えることです．例えばカレンダーの同期，データの共有，システムの動的更新，システム開発過程の一貫性の保持，進化可能なソフトウェアの開発，システムの相互運用などが考えられます．一方，実際に双方向変換を応用すれば上のようなことも出来るけれど，どうやって応用すればよいかわからな

Introduction to Principles and Practice of Bidirectional Transformations.

Hiroyuki Kato, Zhenjiang Hu, Soichiro Hidaka, 国立情報学研究所, National Institute of Informatics.

Kazutaka Matsuda, 東京大学, The University of Tokyo.

コンピュータソフトウェア, Vol.31, No.2 (2014), pp.44–56.
[解説論文] 2013 年 12 月 27 日受付.

い人が多いようです.

双方向変換に関する研究は理論的なものがあれば,実践的なものも数多くあります.今回の「高談闊論」を通じて,普通の人でも読めばわかるように,プログラミング言語,ソフトウェア工学,データベースの視点から,双方向変換の歴史,基本原理,実践,応用,そして今後の課題について話し合いましょう.

2 歴史について

司会: 双方向変換は 30 年ほど前にデータベースの世界で盛んに研究されていたビュー更新問題と深く関わりますよね.

DB: データベースの世界では双方向変換に対応する研究として,ビュー更新問題があります.ビューとはデータベースに格納されているデータ (他のビューも含む) から抽出されたデータのことです.ビューに対する更新を元のデータへと反映させる,ビュー更新問題は,実用上の観点から自然な要求としてあがってきた概念であり,関係データベースの出現とほとんど同時に研究がはじまった比較的古い問題でした [8] が,概念の形式化が確立し体系化が進んで多くの成果が生み出されたのは 80 年代に入ってからです.

データベースビューには,ビューへの問合せ評価時にビュー定義を評価する仮想ビュー (virtual view) と,あらかじめビュー定義を評価しておく実体化ビュー (materialized view) の二種類があります.ビュー定義が複雑になるに従って,ビューへの問合せを効率的に評価するために実体化ビューが必要とされています.ここで話題にしている双方向変換に対応するのは実体化ビューの更新問題ですが,データベースの世界では仮想ビューに対するビュー更新問題がそもそもの始まりです.日本では,Masunaga がいち早くこの問題に取り組み,仮想ビュー更新問題に関して,ビュー定義の意味を考慮に入れることで更新可能なメカニズムの導入に成功しています [37].

司会: データベースには技術的に完成度の高い問合せ言語 (SQL) があるので,その問合せ言語自身を双方向化しないと意味がないという風潮がありますね.80 年代は盛んに研究が行われましたが,90 年後半以降はそれほど多くはないような印象があります.難しい問題だけが残されたのではないかと思います.

PL: データベースの分野とは対照的に,プログラミング言語の分野では,最近の 10 年間に多くの研究がなされています.その先駆けとなる仕事として,ペンシルベニア大の Benjamin Pierce のグループの開発した初めての双方向変換言語 lens があります.この成果は,まず 2004 年の PLAN-X [22] で発表され,その発展版が 2005 年の POPL [18] で発表されています (その後,ジャーナル版がプログラミング言語分野のトップレベルジャーナルの 1 つ TOPLAS [19] に掲載された).彼等のアイデアは,基本的な双方向変換を,双方向変換結合子 (combinator) により組み合せてより大きな双方向変換を構成していくことです.すなわち,構成的な双方向変換の作成です.こうしたアプローチにより,双方向変換が簡単に構成でき,また「よい性質」(後述) が成り立つことを保証することができます.

司会: こういった双方向変換言語はビュー更新問題への 1 つのアプローチだと思います.Pierce のグループは,以前にファイル同期アプリケーション unison を開発したことがあり,こうした研究はその延長にあります.

SE: ソフトウェア工学で双方向変換に取り組んでいて,時として違うコミュニティ,領域として認識されるのはグラフ変換のコミュニティです.というのも,ソフトウェア工学分野での様々なアーティファクト (成果物) は,フローチャートのような古典的なものからグラフで表現されており,グラフ上の操作が必要とされてきた経緯があるからだと思われます.その中でグラフ変換に対するグラフ文法に基く代数的アプローチが 70 年代に提案されています [12].ただ,明示的に双方向変換を意識したものが登場するのは,90 年代のようです [45].

それから,双方向変換の記述方式ですが,標準化団体による規格 QVT (Query/View/Transformation Specification [44]; このなかで双方向変換に直接関係するのは QVT-Relational) が成立したことは注目に値します.

モデル駆動工学での,たとえばモデル駆動開発では,設計段階での抽象的なアーティファクトから具体

的なアーティファクトまでをグラフで表現されるようなモデルとして捉えて，モデルをモデル変換を通して詳細化して実際に動くソフトウェアに持っていこうというアプローチは伝統的にあります．その流れの中でモデル変換の双方向化として双方向変換が研究されてきました．他の分野との大きな違いは，双方向変換の基本単位が一般には関数になっていない，つまりビューのように一方 (データベース) から他方を完全に一意に再現するとは限らないことです．最近の論文 [52] に登場する例でもそうですが，抽象的なものから具体的なコードを生成する時に，出来たコードへの修正の中には他方に反映しない修正も含まれる，つまり同期したい部分への修正だけでなくそれ以外の修正も考慮するといった問題設定です．モデルからコードの最終形を変換で一気に作ってしまうという想定では必ずしもないようです．そして，同期対象部分同士の具体的な対応関係自体をトレースとして独立したアーティファクトと意識するのも特徴のように思われます．

司会: 基本的に対応するリンクをとるんですね．

PL: 同期したい部分だけを抜きだすというのは，先程挙げた Benjamin Pierce のグループの研究の目的の 1 つです．2 つのデータがあったときに，同期したい部分を抜きだして同じ形式のデータに変換してしまえば，その 2 つの変換後のデータ間の同期はたとえば unison の技術を応用するなど他の方法で解決できます．そういった，共通のデータ形式を双方向変換で書けば，異なる形式のデータ間の同期が実現できることになります．

司会: ソフトウェア工学やグラフ変換の世界は実用駆動というか，やりたい事に対する明確なイメージがあって，それをどうシステマティックにやっていったらよいかという方向で発展してきていますね．問題が明確というか現実的な問題を解決するために考えられているように思えます．

3 基本原理について

司会: 双方向変換の原理の議論に移りたいと思います．データベースの分野で 80 年代のはじめ頃に，ビュー更新問題の定式化について多くの研究がされていましたね．

DB: そうですね．その頃，Bancilhon と Spyratos による constant complement を用いた関係ビュー更新の意味の定式化[4]，Dayal と Bernstein による関係ビューに対する更新をどのように元データに伝播させるかの定式化[10]，Keller によるビュー更新を元データに伝播させる際の曖昧さの解決に，ユーザの意図を反映させる方法の提案[33] をはじめ，研究成果が多く発表されました．冒頭でも言いましたがデータベースビューには仮想ビューと実体化ビューの二種類があります．仮想ビューに対する更新を考えるときは operation-based なアプローチを取るのが自然です．一方，実体化ビューに対する更新を考えるときには state-based なアプローチも適用できますね．データベース分野でのビュー更新の成果は constant complement [4] が 1 つのキーになっています．これはビュー定義を，補関数と併せて単射となるようにすることで副作用の無い更新を可能とする枠組みです．

少し具体的に説明します．データベース db からビューを生成するビュー定義を query とすると，ビューと補助データを生成する関数を

$$f\ db = (\text{query}\ db, \text{aux}\ db)$$

のように定義します．関数 f が単射のとき，aux を query の補関数といい，f の逆関数を f^{-1} とします．補関数により生成される補助データは常に一定 (constant) であると仮定すると，次の式で更新されたビュー v と古いデータベース db から新しい db' が計算できるようになります．

$$db' = f^{-1}\ (v, \text{aux}\ db)$$

司会: この constant complement の枠組みでは補関数によっていろいろな更新戦略を定義することができ，理論的に有力な手法のようですが，実際に「よい」補関数を与えるのは容易ではありません．ところで，ビュー更新のときに，ソースのデータベースの (補関数が捕捉する情報以外の) 情報を利用しても大丈夫でしょうか．

DB: データベースビューには open view というデータに関するビューと closed view というメタデー

タに関するビューの二種類があります．データの更新に対して，メタデータの更新は制限されたものとなるので，update strategy が異なってきます[25]．open view におけるビュー更新では，ソースの参照が可能なのに対して closed view においては，ソースの参照が制限されます．

また，ビュー更新についての良い性質として，consitency と acceptablity という性質の定義も比較的早い時期にされており[4]，のちにプログラミング言語の世界で，PutGet と GetPut と呼ばれるようになったようです．

司会: プログラミング言語の分野では，どのように定式化されてきたのでしょうか？

PL: プログラミング言語の分野では，データベースの分野で議論された性質に基づき，双方向変換プログラミング言語設計を行っています．たとえば，前述の結合子に基づくアプローチでは,「よい性質」(後述) を満たす双方向変換を組み合わせることによって,「よい性質」を満たすより大きな双方向変換を構成していきます．

もうすこし具体的に説明すると，ソース S とビュー V の間の双方向変換プログラム (lens と呼ばれる)l は，普通の変換 (get と呼ばれる) としての意味 $[\![l]\!]_{\mathrm{G}} :: S \to V$ と，ビューに対する変更をソースへ反映する変換 (put と呼ばれる) としての意味 $[\![l]\!]_{\mathrm{P}} :: S \times V \to S$ をもっています (図 1)．ここで，put は get の「逆変換」のようなものですが，一般に get は単射とは限らないのでその情報を補うために元のデータ S も引数にとっています．

司会: なるほど，put はビューからソースへの関数ではなく，ソースとビューの組からソースへの関数なので，get が単射でない場合に get の計算により失った情報を復元できるようになっているのですね．このことが味噌なんですね．

PL: そうです．たとえば，ソース $A \times B$ とビュー A 間の原子的な双方向変換プログラム fst は以下のような get と put の意味を持つプログラムだと定義されます．

$$[\![fst]\!]_{\mathrm{G}}\ (a, \_) = a$$
$$[\![fst]\!]_{\mathrm{P}}\ (\_, b)\ a = (a, b)$$

司会: 「原子的」ということは，こういった原子的な双方向変換は，大きな双方向変換の要素となるのでしょうか？

PL: はい．こうした，原子的な双方向変換プログラムを，結合子により合成することで大きな双方向変換プログラムを作成します．たとえば，こうした結合子の 1 つに関数合成 $\circ$ があります．この結合子は，ソース B とビュー C 間の双方向変換プログラム l_1 とソース A とビュー B の双方向変換プログラム l_2 から，それを合成したソース A とビュー C 間の双方向変換プログラム $l_1 \circ l_2$ を定義します．このプログラム $l_1 \circ l_2$ の意味は

$$[\![l_1 \circ l_2]\!]_{\mathrm{G}} = [\![l_1]\!]_{\mathrm{G}} \circ [\![l_2]\!]_{\mathrm{G}}$$
$$[\![l_1 \circ l_2]\!]_{\mathrm{P}}\ (a, c) = [\![l_2]\!]_{\mathrm{P}}\ (a, ([\![l_1]\!]_{\mathrm{P}}\ (b, c)))$$
$$\text{where } b = [\![l_2]\!]_{\mathrm{G}}\ a$$

で定義されます (図 2)．こうした結合子については，合成される双方向変換が「よい性質」を満たしているならば合成結果も「よい性質」を満たしていることが保証されています．なので，結合子に基づいて双方向変換プログラムを作成すると，その構成から,「よい性質」を満たしていることが保証された双方向変換プログラムを得ることができます．

司会: 「よい性質」にはどのようなものがあるのでしょうか？

PL: まずは，GetPut [19] や，acceptability [4] と呼ばれている性質で,「変換後のデータが更新されな

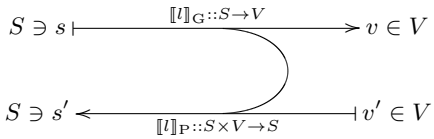

図 1 双方向変換プログラム l

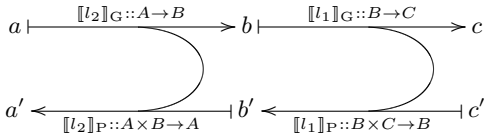

図 2 双方向変換プログラム l_1 と l_2 の合成 $l_1 \circ l_2$

いときには元のデータは更新されない」ことを表現しています.

$$put\ (s, get\ s) = s \qquad \text{(GetPut)}$$

また，PutGet [19]，もしくは consistency [4] と呼ばれる性質も重要です.

$$put\ (s, v) = s' \neq \bot \Rightarrow get\ s' = v \qquad \text{(PutGet)}$$

直感的には，PutGet は，*put* が正しいこと，つまり「*put* は，ビューに対する更新を，ソースに対する更新で (成功すれば) 実現する」ことを表現しています. 関数 *put* は一般には部分関数かもしれないことに注意して下さい．以上の 2 つの性質，GetPut や PutGet が双方向変換においてもっとも重要な性質です.

これらの「よい性質」の他に,「望ましい性質」についても議論されています．たとえば，以下の PutPut は「望ましい性質」の 1 つです.

$$put\ (s, v) = s' \neq \bot \Rightarrow put\ (s, v') = put\ (s', v') \qquad \text{(PutPut)}$$

直感的には，PutPut は,「(一連の *put* の適用においては) 更新後のビューが決まれば，更新後のソースが決まる」ことを表現しています．このことの利点としては,「ビュー上の更新の反映結果が，反映のタイミングに依存しない」ことと「ビューの上でのアンドゥ操作が，ソースの上でのアンドゥ操作に対応する」ことが挙げられます．データベースの世界で議論されてきた constant complement は，GetPut，PutGet および PutPut の 3 つすべてを満たすことと等価であることが知られています [4].

司会: 双方向変換システムはすべての「望ましい性質」を満たす必要があるのでしょうか？

PL: いいえ，それは双方向変換システムの使用する目的によりけりです.

DB: closed view に対しては，PutPut が要請されることが多いようです [4] [25].

PL: しかし，この PutPut を満足させるために，*put* が特定の更新を反映できないことがあること (例：*get* が *mapFst* である場合を考えよう (図 3)．ソースが $[(a, b)]$ である場合に，ビューを $[a]$ から [] に更新することを許すと，再び $[a]$ に戻したときに b の情報が手に入らず，PutPut を満たせない．そのため，PutPut を満たす双方向変換では，ビューを $[a]$ を [] に更新することは難しい) が知られています [34]．そのため，多くのシナリオで，PutPut は強すぎる要求であると考えられています．PutPut のより弱いバージョンについてはいくつか議論はありますが [21] [30]，決定的なものはまだ出ていない印象です.

```
mapFst []       = []
mapFst ((a,b):x) = a:mapFst x
```

図 3　get プログラムの例

司会: ここまで説明してきた結合子に基づく方法は原子的な双方向変換と結合子によって性質のよい双方向変換システムを構築することですが，順変換 get からよい性質を満たすような逆変換 put を自動的に導出できると便利ですね.

PL: それは (自動) 双方向化 (bidirectionalization) といいます．双方向化では，ユーザに get を (制限はあるものの) 一般的なプログラミング言語で書いてもらい，あとはシステムが「よい」put を自動で導出します [38] [49] [39]．たとえば，[38] のアプローチでは図 3 のような get プログラムに対し，図 4 の put プログラムと等価なプログラムを自動で導出することが可能です．また，[49] では，多相的な get 関数を取り，対応した put 関数を返す関数

$$bff :: (\forall \alpha. F\ \alpha \to G\ \alpha) \to F\ \beta \to G\ \beta \to F\beta$$

が提供されています．ここで，F と G は，大雑把に言えば，コンテナ (例：リスト，木) をあらわす型構成子です．こうしたアプローチでは，得られた put プログラムが，入力となる get プログラムに対して「よい性質」を満たすことが，Free Theorem [50] により保証されています．[38] の手法も [49] の手法も，原理的には constant complement に基づいています [20].

司会: 言語設計においては，こうした「よい性質」

```
mapFstP []       []  = []
mapFstP ((_,b):x) a:y = (a,b):mapFstP x y
```

図 4　図 3 のプログラムに対応する put プログラムの 1 つ

を保ちつつ，ユーザにとってわかりやすい言語を如何に提供するかが，課題となっています．

PL: これまで紹介したアプローチの他に「関係」を書いて，それから双方向の変換を導出するアプローチもあります．たとえば，biXid [32] や XSugar [6] はこの分類です．この biXid は異なる形式の XML 文書の相互変換を目的しており，XSugar はあるデータの XML 表現と「古典的」なフラットなテキスト表現との相互変換を目的としています.「よい性質」としては，biXid では特に何も要求されていませんが，XSugar では同じデータの表現間の変換を目的としているため変換が全単射であることが要求されています．これら biXid および XSugar は，技術的には，自然言語処理分野で機械翻訳に利用される同期文法 (synchronous grammar) [1] との関連が深く，これまで紹介した技術とは原理が異なっています．

司会: ここまで，データベースやプログラミング言語の分野で双方向変換システムの構成方法を紹介しました．グラフ変換とソフトウェア工学の分野はどうでしょうか．double pushout と Triple Graph Grammar (TGG) という 2 つの難しい言葉はよく耳にしますが，それらはどういうことでしょうか．

SE: 双方とも 2 つのグラフの間の対称的な双方向変換に使われるグラフ文法に基づく枠組みで，しばしば圏論と複雑な線図を使って説明されるため，難しいという印象を持たれているようです．2 つのアプローチは少し系譜が異なりますが†1，TGG の様々な性質の議論のために double pushout が利用されることもあります．まず，比較的単純な double pushout から説明します．

double pushout では，圏論の pushout (押し出し) という概念を用います．グラフを対象としグラフ morphism を射とする圏を考えます．変換は，そのような圏において pushout となる可換な線図を 2 つ接続した図 (図 5) によって表されます [12]．double pushout の名はこうした 2 つの pushout に由来しています．$\grave{p}$, m, e, $\grave{\tilde{p}}$ からなる線図と p', m', e, $\tilde{p}'$ からなる線図がそれぞれ pushout となっています．G は変換元 (ソース) のグラフ，H は変換先 (ターゲット) のグラフです．上側の辺 $L \leftarrow K \rightarrow R$ は個々のグラフ変換ルールを表し，順変換の際には G からグラフ (パターン) L を matching morphism m を通して抽出し，(その部分グラフを) グラフ R で置き換えることによりグラフ H を生成します．つまり，グラフの書換え系となっています．上記変換ルールを生成規則とすると，一点グラフを始点とするグラフ文法を構成するため，graph grammar とも呼ばれています．グラフ変換から一旦離れて pushout の construction そのものを考えると，例えば図 5 の左側の線図では $K, \grave{p}, L, e, C$ が与えられたときに G(と m,$\grave{\tilde{p}}$) を作ります．しかし図 5 によるグラフ G の変換に立ちもどると，$K, \grave{p}, L, m, G$ が与えられているので，pushout complement により C (と e,$\grave{\tilde{p}}$) を一旦つくり (C はグラフ G をルール $L \leftarrow K \rightarrow R$ に沿って変換する際の中間段階)，右半分は今作った C と，K, e, p', R を使って通常の pushout construction により H を生成します．逆変換も同様な過程を逆向きに行うことにより達成します．

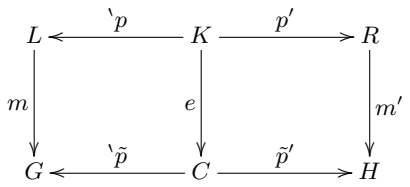

図 5 Double Pushout

司会: 双方向変換との関係をもう少し説明いただけますでしょうか．

SE: double pushout に近い方法で双方向変換を目指すものも最近出て来てはいます [35] が，double pushout を直接利用する方法は，実は双方向変換というよりは，可逆変換と呼ばれているようです [9]．ですので次に，双方向変換としてより広く用いられている TGG [45] を説明します．このグラフ文法は，対称的である点，ソースとターゲットの対応を $L \leftarrow K \rightarrow R$ のような三つ組 (triple) で表現する点では，double pushout と同じですが，グラフ文法の観点では，pair grammar の流れをくんでおり，ソースグラフからターゲットグラフの生成，またはその逆を直接行うので

†1 詳しい対応関係に御興味のある方は [14] を御覧下さい．

はなく，ソース，ターゲット両方のグラフの一部と，その間のリンクを表現する中間グラフの 3 つを組合せたグラフをひとかたまり (triple graph) として扱い，そのかたまりを生成規則により成長させていく点が異なります．TGG としての変換ルールは，三つ組グラフパターンの，別の三つ組グラフパターンへの置き換えで表現されます．このような三つ組を単位としたグラフ文法に基づくことから，Triple Graph Grammar と呼ばれています．pushout の使われ方としては，三つ組グラフパターン L をもとに，それを別の三つ組グラフパターン R に書き換える graph morphism と，L を作りかけの三つ組グラフ G にマッチさせる matching graph morhism を使って，一度だけ pushout を使い一気に書き換え後の三つ組グラフを生成します．

司会: なるほど，double pushout との理論的な違いは分かりましたが，では，実際にどうやって双方向にグラフを変換するのですか．

SE: 実際にソースからターゲットを生成するには，三つ組を並行して成長させるルールを直接使わず，ソース側だけを変換対象のグラフになるまで成長させるソースルール (中間グラフ，ターゲットグラフは空のまま) と，完成したソースグラフにあわせて中間グラフ，ターゲットグラフを成長させていく順方向 (forward) ルールの列に分解し，それらを順に適用します．逆変換も同様の手順で行います．

司会: データベースやプログラミング言語分野で行われてきたアプローチと随分趣が違うようですが，TGG は双方向変換としてはどのような性質を満たすように設計されているのですか．

SE: まず，グラフ文法では文脈依存の書き換えにより非常に表現力が高くなっており，扱いやすい性質を満たすための，双方向変換の性質以外の性質が多く議論されます．たとえば，completeness [15] は，ある種類のターゲットのグラフを作りたいと決めたらこのルールがあればすべて作ることができるというような，全射のようなものでしょうか．それとすべての情報がソースからターゲットに伝えられるかどうか (information preserving [13])．view update だと一般にはターゲットにはソースの情報の一部しか現れないという使い方だと思いますが，そうではなくすべての情報が他方に伝えられるかどうかに着目するものです．

さて，双方向変換としての性質ですが，対称的な枠組では，順変換でも追加の引数 (古いターゲット) を取ります．ソースの一部とターゲットの一部のみを部分的に同期するのに使うことができます．ですので，少し異なる表現になりますが，まず GetPut や acceptability に対応するものは hippocraticness, PutGet や consistency に相当するものは correctness と呼ばれています [46]．式で表現すると，ソース s とターゲット t の間に一貫性があるとき $(s,t) \in R$, 順変換を $\overrightarrow{T}$，逆変換を $\overleftarrow{T}$ とすると，

$$(s,t) \in R \Rightarrow \overrightarrow{T}(s,t) = t \wedge \overleftarrow{T}(s,t) = s \quad \text{(hippocraticness)}$$

$$(s, \overrightarrow{T}(s,t)) \in R \wedge (\overleftarrow{T}(s,t), t) \in R \quad \text{(correctness)}$$

となり，TGG はこれらの性質を満たします [14] [27]．ただし，変換が上記のような関数で表現出来るようにするためには，TGG などのルールベースのアプローチでは変換が決定的になるルールのみに限定することが前提になります．

PutPut に相当する強い性質は，invertibility と呼ばれ，TGG がこれを満たすための条件も知られています [27]．本稿でも先にこれ等の性質をすべて満たすことが求められているかという議論がありましたが，SE の分野でも同様の議論があり [11]，[27] では TGG における invertibility を弱めた性質である weak invertibility も議論されています．直観的には，往復変換前後で値がずれてしまう，つまり往復すると同じ値に戻れないけれど，ずれの前後の値は，それぞれ他方に伝播すると同じ値になるというものです．

4 支援環境について

司会: 双方向変換システムの開発を支援するためのツールと環境について，ご紹介していただきたいと思います．

DB: ほとんどの商用の関係データベース (Oracle, IBM DB2, MySQL, Microsoft SQL sever など) は，限定的ではありますが (仮想，実体化双方の) ビュー

更新を実現しています.

司会: 具体的にどのような制限がありますでしょうか.実用になっていますでしょうか.

DB: ビュー上の組と元データ上の組が一対一に対応しており,かつ元データ及びビュー作成時の制約(関数従属性など)を満たす場合においてのみ可能なように制限されているようです.

司会: プログラミング言語の分野にはどのようなツールや支援環境があるのでしょうか?

PL: 双方向変換言語 lens については,様々な実装があります.代表的なものは,結合子による双方向変換構成を提案したグループによる,双方向プログラミング言語 Boomerang [5] でしょうか†2.この処理系は,テキストデータの双方向変換を主眼としたものです.また,他にもライブラリとしての実装が多数存在しています.たとえば,Haskell のライブラリだと,`lens` パッケージ†3がもっとも有名でしょうか.また,双方向プログラミング言語の実装としては,NII のグループによる GRoundTram システムがあります†4.この処理系は,ソフトウェア開発における応用を目指し,グラフデータに対する双方向変換を目的としています.双方向化のシステムについても実装があります.たとえば,`b18n` システム†5では,特定の制限されたプログラミング言語で記述された get のプログラムから,put のプログラムを自動導出してくれます.前述の多相的な get 関数から put のプログラムを導出するアプローチはライブラリとして実装・提供されています†6.このライブラリを用いれば,多相的な get に限定されているものの,Haskell の中で完結しているので気軽に双方向化を利用することができます.同様のアイデアに基づくシステムでは,「多相的」という制約が緩められているもの [39] があり,こちらもライブラリが公開されています†7.前述の biXid についてもソースコードが公開されていますし†8,XSugar についても処理系が公開されています†9.また,FliPpr というシステム†10では,ユーザが書いた整形出力器 (pretty-printer) を与えると,対応した構文解析器を出力してくれます [40].

司会: ソフトウェア工学の分野ではどうでしょうか.

SE: TGG を使った Moflon [2]†11 のようなしっかりとしたツールがあり,入力をグラフィカルにできる点も非常に魅力的です.もうひとつの特徴は,ルールが互いに干渉するかどうかを教えてくれる等のサブツールが充実していることです [26] [24] [47].先ほど,TGG ではルールがセットで提供されていると述べましたが,その干渉を手で補修するのは大変な作業です.そこで,どのルールとどのルールに干渉が起こっているかを,行列形式で示してくれるのです.このようなサポートはとても現実的で,非常に強力であると言えます.

先に述べた標準化された変換言語 QVT についても実装があります.ikv++社による medini QVT [41] は subset†12を実装しています.De facto 標準といってもよいモデル変換言語 ATL [31] をバイトコードレベルで双方向化した SyncATL [52] もあります†13.しかし,挿入を伴う更新が逆変換で伝播できないといった制限があるようです.

司会: どの分野でも,色々なツールと環境は開発されていますが,まだ研究段階と言ってもよいと思います.企業で使える実用的な環境の開発が必要です.

5 応用について

司会: 双方向変換について,GRACE セミナーの報告論文 [9] やダグストゥール (Dagstuhl)・セミナーのサーベイ論文 [29] にも様々な応用が書かれていますが,それぞれの分野で印象的な応用と言えば何があり

†2 http://www.seas.upenn.edu/~harmony/
†3 http://hackage.haskell.org/package/lens
†4 http://biglab.org/ より入手可
†5 http://www-kb.is.s.u-tokyo.ac.jp/~kztk/b18n/
†6 http://hackage.haskell.org/package/bff
†7 https://bitbucket.org/kztk/cheap-b18n/
†8 http://mayah.jp/article/2006/bixid
†9 http://www.brics.dk/xsugar/
†10 http://www-kb.is.s.u-tokyo.ac.jp/~kztk/FliPpr/
†11 http://www.moflon.org
†12 例えば checkonly mode は実装されていないようです [36].
†13 http://sei.pku.edu.cn/~xiongyf04/modelSynchronization.html

ますでしょうか？

PL: 設定ファイル編集システム Augeas[†14]には，lens が応用されています．Augeas を用いることで，`/etc/`以下の設定ファイルの内容を，まるで木構造であるかのようにパス式でたどることができ，また更新ができます．この，設定ファイルと木構造の間の対応は，lens によって記述されています．Augeas は様々な Linux ディストリビューションのパッケージ管理システムで提供されているので，入手・インストールが容易になっています．東大の PSD プロジェクト[†15]の成果物である双方向変換に基づく web ページ管理システム Vu-X も応用例のひとつです [42]．このシステムでは，ユーザがウェブページをブラウザ上で編集することが可能であり，双方向変換を通じてページ内/ページ間の一貫性を保証しています．

DB: ビュー更新以外のものとして，実体化ビューを用いた問合せ処理 (answering queries using materialized views) があります．元データに対する問合せを元データではなくビューを使って処理するという問題で，最適化のみならずこれ自体に多くのデータベース応用があります [23]．振る舞いの良い *put* 関数が定義できれば，この問題の解決に繋がります．関係データモデルだけでなく XML などの木構造を対象に研究が進んでいる状況です [51]．

また，異なるスキーマのもとに存在している同じ意味を持つデータに対する統一的なアクセスを目標とするデータ相互運用問題 (the data interoperability challenge) において，スキーマの違いはスキーママッピングで記述しますが，近年このスキーママッピングの逆変換に関する研究が活発になってきています．特にスキーママッピングが関係で表わされている場合についての研究がいくつかあり [16] [3] [17]，双方向変換の技術と密接な関連があります．

SE: モデル駆動開発においては，双方向変換は，特にラウンドトリップエンジニアリング，つまりモデルとそれから生成されたコードへの変更を双方向へ伝播する手段としては良く知られています．実際，最終的に生成されたコードを実行して，露見した問題があった場合，その修正をなるべく上流の工程に反映させたいという要求は耳にします．ただ，実際には思った程簡単ではないようです．ICSE'12 で発表されている応用例 [53] では，API 部分の同期は既存のものを使い，関数 (メソッド) 本体そのものについては双方向変換を使うという，ハイブリッドな方法がとられています．もうひとつ印象的なものは，異なるドメインの間の橋渡しへの応用です．例えば，あるシステムを設計し，検証ツールでその性質を形式的検証する場合，システムの記述を検証ツールが解釈可能な言語に翻訳する必要があります．しかし，そこで出た反例などのエラーメッセージは必ずしも設計者に分かりやすいものではありません．そこで従来は設計者が分かるような表現への逆変換，つまりシステム設計ドメインへのフィードバックを別途記述していました．この 2 つの変換に双方向変換を使えば，単一の記述で，翻訳とフィードバックが一度に実現できます．

6 今後の課題について

司会: 双方向変換に関する研究は歴史が長く，様々な観点から研究が進んでいますが，まだ課題が多くなかなか実用化できていないのが現実です．最後に，それぞれの分野でどうしても解決したい課題を紹介していただきたいと思います．

PL: 双方向プログラミング言語の設計については課題がまだまだあります．そのひとつは，扱えるデータ構造についてです．木や文字列に対してはさまざまな議論がありますが，グラフ，高階抽象構文木などの複雑なデータ構造については—グラフについては [28] に議論がありますが—まだ十分な議論がなされていない状態です．別の課題は，どの *put* がほしいのかを，ユーザがどういった形で記述するかについてです．一般には，1 つの *get* に対し，「よい性質」を満たす *put* は複数あります．なので，「どの *put* がほしいのか」を記述する「適切な」手段をユーザに与えることが課題となっています．

DB: データベースの分野固有の課題ではありませんが，何ができて何ができないのか，そして，できるためにはどういう情報があればどこまでできて，でも，どんな情報があってもこれ以上はできないのか，

†14 http://augeas.net/

†15 https://www.psdlab.org/

という境界線を示していくことが必要とされています，そのための方法論として，特にデータベースの分野では data provenance [48] と双方向変換の融合が挙げられます [7]．また，新たな応用として，ビッグデータ処理における双方向変換などが挙げられると思います．

SE: これも分野横断的課題ですが，双方向変換に関する諸性質の概念の共有にはまだ時間を要しているようです．また，双方向変換ツールはそれぞれ固有の動機に基づき設計されているため，形式的な性質を満たしているかどうかの共通の判断基準 (ベンチマーク) を作る試みも進められています．

双方向変換における非決定性への対処については，モデル変換でも課題となっています．先程の複数の *put* からの選択に対応しますが，最も変更が少ないものを選ぶというのも 1 つの妥当な基準です．イギリスでは双方向変換における最小変更 (least change) 理論に関するプロジェクト研究が進行中です†16．特にモデルのような複雑な構造における最小変更の基準には議論が必要と思われます．グラフ編集距離を基準とした最小変更を Alloy のソルバーを使って実現している例がありますが [36] [43]，双方向変換のデバッグと検証に有用であるものの，大規模なモデルへの適用は今後の課題となっています．

7 まとめ

司会: 双方向変換の研究は横断的です．研究者がそれぞれの分野で「独創的」研究を行うことだけではなく，分野を超えた研究協力も大変重要です．そのため，2008 年に日本の湘南で，2010 年にドイツのダグストゥールで，2013 年にカナダのバンフ (Banff) センターで各分野で活躍されている研究者が集まり，双方向変換に関する研究交流を行い，挑戦的な課題について議論を深めてきました．また，双方向変換に関する国際ワークショップ (BX Workshop) もこれまで ETAPS の併設イベントとして 2 回行われ，次回は EDBT 2014 の併設ワークショップとして開催される予定です．今後，双方向変換，さらに双方向計算，双方向プログラミング等が魅力的な新しい研究分野としてもっと注目されると信じております．

†16 A Theory of Least Change for Bidirectional Transformations, http://www.cs.ox.ac.uk/projects/tlcbx/

参 考 文 献

[1] Aho A. V. and Ullman. J. D.: Syntax directed translations and the pushdown assembler, *J. Comput. Syst. Sci.*, Vol. 31, No. 1(1969), pp. 37–56.

[2] Amelunxen, C., Königs, A., Rötschke, T. and Schürr, A.: MOFLON: A Standard-Compliant Metamodeling Framework with Graph Transformations, in *Model Driven Architecture - Foundations and Applications: Second European Conference*, Rensink A. and Warmer, J. (eds.), Vol. 4066 of Lecture Notes in Computer Science (LNCS), Springer Verlag, 2006, pp. 361–375.

[3] Arenas, M., Pérez, J., Reutter, J. and Riveros, C.: Query language-based inverses of schema mappings: Semantics, computation, and closure properties, *The VLDB Journal*, Vol. 21, No.6(2012), pp. 823–842.

[4] Bancilhon, F. and Spyratos, N.: Update semantics of relational views, *ACM Trans. Database Syst.*, Vol. 6, No.4(1981), pp. 557–575.

[5] Bohannon, A., Foster, J. N., Pierce, B. C., Pilkiewicz, A. and Schmitt, A.: Boomerang: resourceful lenses for string data, in *POPL*, Necula, G. C. and Wadler, P. (eds.), ACM, 2008, pp 407–419.

[6] Brabrand, C., Møller, A. and Schwartzbach, M. I.: Dual syntax for XML languages, *Inf. Syst.*, Vol. 33, No. 4–5(2008), pp. 385–406.

[7] Cheney, J.: Bidirectionality, traceability and provenance, Bi-directional transformations(BX)–Theory and Applications Across Disciplines, 2013, Banff International Research Station.

[8] Codd, E. F.: Recent investigations in relational data base systems, in *IFIP Congress*, 1974, pp. 1017–1021.

[9] Czarnecki, K., Foster, J. N., Hu, Z., Lämmel, R., Schürr, A. and Terwilliger, J. F.: Bidirectional transformations: A cross-discipline perspective, in *ICMT*, Paige, R. F. (ed.) Vol. 5563 of Lecture Notes in Computer Science, Springer, 2009, pp. 260–283.

[10] Dayal, U. and Bernstein, P. A.: On the correct translation of update operations on relational views, *ACM Trans. Database Syst.*, Vol. 7, No. 3(1982), pp. 381–416.

[11] Diskin, Z., Xiong, Y., Czarnecki, K., Ehrig, H., Hermann, F. and Orejas, F.: From state- to delta-based bidirectional model transformations: The symmetric case, in *Model Driven Engineering Languages and Systems*, Whittle, J., Clark, T. and Kühne, T. (eds.), Vol. 6981 of Lecture Notes in Computer Science, Springer, 2011, pp. 304–318.

[12] Ehrig, H., Pfender, M. and Schneider, H. J.:

Graph-grammars: An algebraic approach, in *Switching and Automata Theory, 1973, SWAT '08, IEEE Conference Record of 14th Annual Symposium on*,1973, pp. 167–180.

[13] Ehrig, H., Ehrig, K., Ermel, C., Hermann, F. and Taentzer, G.: Information preserving bidirectional model transformations, in *Proceedings of the 10th international conference on Fundamental approaches to software engineering*, FASE'07, Springer-Verlag, 2007, pp. 72–86.

[14] Ehrig, H., Ermel, C. and Hermann, F.: On the relationship of model transformations based on triple and plain graph grammars, in *Proceedings of the Third International Workshop on Graph and Model Transformations*, GRaMoT '08, ACM, 2008, pp 9–16.

[15] Ehrig, H., Ermel, C., Hermann, F. and Prange, U.: On-the-fly construction, correctness and completeness of model transformations based on triple graph grammars, in *Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems*, MODELS '09, Springer-Verlag, 2009, pp. 241–255.

[16] Fagin, R., Kolaitis, P. G., Popa, L. and Tan, W. C.: Reverse data exchange: Coping with nulls, *ACM Trans. Database Syst.*, Vol. 36, No. 2(2011), 11:1–11:42.

[17] Fagin, R. and Nash, A.: The structure of inverses in schema mappings, *J. ACM*, Vol. 57, No. 6(2010), 31:1–31:57.

[18] Foster, J. N., Greenwald, M. B., Moore, J. T., Pierce, B. C. and Schmitt, A.: Combinators for bi-directional tree transformations: a linguistic approach to the view update problem, in *POPL*, Palsberg, J. and Abadi, M. (eds.), ACM, 2005, pp 233–246.

[19] Foster, J. N., Greenwald, M. B., Moore, J. T. Pierce, B. C. and Schmitt, A.: Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem, *ACM Trans. Program. Lang. Syst.*, Vol. 29, No. 3(2007), 17:1–17:65.

[20] Foster, N., Matsuda, K. and Voigtländer, J.: Three complementary approaches to bidirectional programming, in *Generic and Indexed Programming*, Gibbons, J. (ed.), Vol. 7470 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 1–46.

[21] Gottlob, G., Paolini, P. and Zicari, R.: Properties and update semantics of consistent views, *ACM Trans. Database Syst.*, Vol. 13, No. 4(1988), pp. 486–524.

[22] Greenwald, M. B., Moore, J. T., Pierce, B. C. and Schmitt, A.: A language for bi-directional tree transformations, in *PLAN-X*, 2004, pp. 83–95.

[23] Halevy, A. Y.: Answering queries using views: A survey, *VLDB J.*, Vol. 10, No. 4(2001), pp. 270–294.

[24] Hausmann, J. H. Heckel, R. and Taentzer, G.: Detection of conflicting functional requirements in a use case-driven approach: A static analysis technique based on graph transformation, in *Proceedings of the 24th International Conference on Software Engineering*, ICSE '02, ACM, 2002, pp. 105–115.

[25] Hegner, S. J.: Foundations of canonical update support for closed database views, in *ICDT*, Abiteboul, S. and Kanellakis, P. C. (eds.), Vol. 470 of Lecture Notes in Computer Science, Springer, 1990, pp. 422–436.

[26] Hermann, F., Ehrig, H., Golas, U. and Orejas, F.: Efficient analysis and execution of correct and complete model transformations based on triple graph grammars, in *Proceedings of the First International Workshop on Model-Driven Interoperability*, MDI '10, ACM, 2010, pp. 22–31.

[27] Hermann, F., Ehrig, H., Orejas, F., Czarnecki, K., Diskin, Z. and Xiong, Y.: Correctness of model synchronization based on triple graph grammars, in *Proceedings of the 14th international conference on Model driven engineering languages and systems*, MODELS'11, Springer-Verlag, 2011, pp. 668–682.

[28] Hidaka, S., Hu, Z., Inaba, K., Kato, H., Matsuda, K. and Nakano. K.: Bidirectionalizing graph transformations, in *ICFP*, Hudak, P. and Weirich, S. (eds.), ACM, 2010, pp. 205–216.

[29] Hu, Z., Schürr, A., Stevens, P. and Terwilliger, J. F.: Dagstuhl seminar on bidirectional transformations (bx), *SIGMOD Record*, Vol, 40, No. 1(2011), pp. 35–39.

[30] Johnson, M. and Rosebrugh, R. D.: Lens put-put laws: monotonic and mixed, in *ECEASST*, 49, 2012.

[31] Jouault, F., Allilaire, F., Bézivin, J. and Kurtev, I.: Atl: A model transformation tool, *Sci. Comput. Program.*, Vol. 72, No. 1–2(2008), pp. 31–39.

[32] Kawanaka, S. and Hosoya, H.: biXid: a bidirectional transformation language for XML, in *ICFP*, Reppy, J. H. and Lawall, J. L. (eds.), ACM, 2006, pp. 201–214.

[33] Keller, A. M.: Choosing a view update translator by dialog at view definition time, in *VLDB*, Chu, W. W., Gardarin, G., Ohsuga, S. and Kambayashi, Y. (eds.), Morgan Kaufmann, 1986, pp. 467–474.

[34] Keller, A. M.: Comments on bancilhon and spyratos' "update semantics and relational views", *ACM Trans. Database Syst.*, Vol. 12, No. 3(1987), pp. 521–523.

[35] Lamo, Y., Mantz, F., Rutle, A. and de Lara, J.: A declarative and bidirectional model transformation approach based on graph co-spans, in *Proceedings of the 15th Symposium on Principles and Practice of Declarative Programming*, PPDP '13, ACM, 2013, pp. 1–12.

[36] Macedo, N. and Cunha, A.: Implementing QVT-R Bidirectional Model Transformations Using Alloy, in *Proceedings of the 16th International*

Conference on Fundamental Approaches to Software Engineering, FASE'13, Springer-Verlag, 2013, pp. 297–311.
[37] Masunaga, Y.: A relational database view update translation mechanism, in *VLDB*, Dayal, U., Schlageter, G. and Seng, L. H. (eds.), Morgan Kaufmann, 1984, pp. 309–320.
[38] Matsuda, K., Hu, Z., Nakano, K., Hamana, M. and Takeichi. M.: Bidirectionalization transformation based on automatic derivation of view complement functions, in *ICFP*, Hinze, R. and Ramsey, N. (eds.), ACM, 2007, pp. 47–58.
[39] Matsuda, K. and Wang, M.: Bidirectionalization for free with runtime recording: or, a lightweight approach to the view-update problem, in *PPDP*, Peña, R. and Schrijvers, T. (eds.), ACM, 2013, pp. 297–308.
[40] Matsuda, K. and Wang, M.: Flippr: A prettier invertible printing system, in *ESOP*, Felleisen, M. and Gardner, P. (eds.), Vol. 7792 of Lecture Notes in Computer Science, Springer, 2013, pp. 101–120.
[41] Medini QVT, http://projects.ikv.de/qvt/.
[42] Nakano, K., Hu, Z. and Takeichi, M.: Consistent web site updating based on bidirectional transformation, in *WSE*, Liu, C. and Ricca, F. (eds.), IEEE Computer Society, 2008, pp. 45–54.
[43] Macedo, N., Guimarães, T. and Cunha, A.: Model repair and transformation with echo, in *Proc. ASE*, IEEE, 2013, pp. 694–697.
[44] Object Management Group: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Version 1.0, 2008.
[45] Schürr, A.: Specification of graph translators with triple graph grammars, in *Graph-Theoretic Concepts in Computer Science, 20th International Workshop, WG '94, Herrsching, Germany*, Mayr, E. W., Schmidt, G. and Tinhofer, G. (eds.), Vol. 903 of Lecture Notes in Computer Science, Springer, 1995, pp. 151–163.
[46] Stevens. P.: Bidirectional model transformations in qvt: semantic issues and open questions, *Software and System Modeling*, Vol. 9, No. 1(2010), pp. 7–20.
[47] Taentzer, G.: AGG: A graph transformation environment for modeling and validation of software, in *AGTIVE*, Pfaltz, J. L., Nagl, M. and Böhlen, B. (eds.), Vol. 3062 of LNCS, Springer, 2003, pp. 446–453.
[48] Tan, W. C.: Research problems in data provenance, *IEEE Data Eng. Bull.*, Vol. 27, No. 4(2004), pp. 45–52.
[49] Voigtländer, J.: Bidirectionalization for free! (pearl), in *POPL*, Shao, Z. and Pierce, B. C. (eds.), ACM, 2009, pp. 165–176.
[50] Wadler, P.: Theorems for free! in *Proceedings of the Fourth International Conference on Functional Programming Languages and Computer Architecture*, FPCA '89, ACM, 1989, pp. 347–359.
[51] Wang, J. and Yu, J. X.: Revisiting answering tree pattern queries using views, *ACM Trans. Database Syst.*, Vol. 37, No. 3(2012), 18:1–18:34.
[52] Xiong, Y., Liu, D., Hu, Z., Zhao, H., Takeichi, M. and Mei, H.: Towards automatic model synchronization from model transformations, in *22nd IEEE/ACM International Conference on Automated Software Engineering* (*ASE 2007*), ACM Press, 2007, pp. 164–173.
[53] Yu, Y., Lin, Y., Hu, Z., Hidaka, S., Kato, H. and Montrieux, L.: Maintaining invariant traceability through bidirectional transformations, in *34th International Conference on Software Engineering*, Glinz, M., Murphy, G. C. and Pezzè, M. (eds.), IEEE, 2012, pp. 540–550.

加藤弘之

1990 年東京理科大理学部物理学科卒業．同年日本ユニシス株式会社入社．1999 年，奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．同年，学術情報センター助手．2001 年，国立情報学研究所コンテンツ科学研究系助教．博士 (工学)．データベース問合せ言語の最適化に興味を持つ．情報処理学会，ACM 各会員．

胡　振江

1988 年中国上海交通大学計算機科学系を卒業．1996 年東京大学大学院工学系研究科情報工学専攻博士課程修了．同年日本学術振興会特別研究員を経て，1997 年東京大学大学院工学系研究科情報工学専攻助手，同年 10 月同専攻講師，2000 年同専攻准教授，2008 年より国立情報学研究所教授．博士 (工学)．プログラミング言語，関数プログラミング，モデル変換とモデル駆動ソフトウエア開発，並列プログラミングなどに興味を持つ．情報処理学会，ACM，IEEE 各会員．

日高宗一郎

1994 年東京大学工学部電気工学科卒業．1999 年同大学大学院電子情報工学専攻博士課程修了．同年学術情報センター助手．現在，国立情報学研究所助教．2002 年 4 月より総合研究大学院大学情報学専攻助教 (併任)．博士 (工学)．言語処理系，データベースプログラミング言語，オペレーティングシステム，文書処理システム等の研究に従事．ACM，電子情報通信学会，情報処理学会各会員．

松 田 一 孝

2004 年東京大学工学部計数工学科卒業．2009 年東京大学大学院情報理工学系研究科数理情報学専攻博士課程修了．2008 年より 2 年間日本学術振興会特別研究員．2010 年より 2 年間東北大学大学院情報科学研究科助教．2012 年より東京大学大学院情報理工学系研究科コンピュータ科学専攻助教．博士 (情報理工学)．関数プログラミング，プログラム変換，領域特化言語，プログラム逆計算等に興味を持つ．情報処理学会，ACM 各会員．