Towards a Complete Picture of Lens Laws

Keisuke Nakano
k.nakano@acm.org
Tohoku University
Sendai, Japan

ABSTRACT

Bidirectional transformation, also called lens, has played important roles in maintaining consistency in many fields of applications. A lens is specified by a pair of forward and backward functions which relate to each other in a consistent manner. The relation is formalized as a set of equations called lens laws. This report investigates precise dependencies among lens laws: which law implies another and which combination of laws implies another. The set of such implications forms a complicated graph structure. It would be helpful to check a well-definedness of bidirectional transformation in a lightweight way.

KEYWORDS

bidirectional transformation, asymmetric lens, lens laws

1 INTRODUCTION

Bidirectional transformation has been called lens after Foster et al. [4] revisited a classic view updating problem introduced by Bancilhon and Spyratos [1]. They have played an important role for maintaining consistency in many fields of applications, database management systems, algebraic data structure interface on programming, and model-driven software development. In particular, lenses are employed in a core foundation of Dejima architecture [8], distributed systems where data are maintained in different peers, which some parts of data in peers are expect to be synchronized.

A lens is a pair of a forward function get and a backward function put which are used for maintaining consistency between two related data, a source and a view. Let S and V be sets of sources and views. The function get : S \rightarrow V generates a view from a given source data typically by extracting a part of the source and arranging it in an appropriate way; the function put : S \times V \rightarrow S reflects an update on the view with assist of the original source because views have less information than the corresponding sources in general.

To define a meaningful bidirectional transformation, two functions, get and put, which forms a lens should relate to each other. The relationship is characterized by equations for these functions called lens laws. Figure 1 shows four typical lens laws introduced in [4]. The (StrongGetPut) law requires that a source can always be determined by put only with an updated view independently of the original source. Under this law, views are as informative as the corresponding sources. The (GetPut) law is a weaker version of the (StrongGetPut) law. This law requires that the same source as original is obtained by put whenever the view has not been updated. The (PutGet) law is about consistency of view updating. This law requires that any updated source by put with an updated view yields the same view by get. The (PutPut) law is a condition imposed only on the put function. This law requires that a source updated twice (or possibly more) by put with different views consecutively is the same as one obtained by put with the last view.

These core lens laws characterize three practical properties on lenses for meaningful bidirectional transformation: bijective, well-behaved, and very-well-behaved. A bijective lens should satisfy the (StrongGetPut) and (PutGet) laws. A well-behaved lens should satisfy the (GetPut) and (PutGet) laws. A very-well-behaved lens should satisfy the (GetPut), (PutGet) and (PutPut) laws. Programmers defining lenses for bidirectional transformation need to select an appropriate property for lenses according to their purpose and application and check if a defined lens satisfies the corresponding lens laws.

One of the solutions is to use domain-specific languages for bidirectional transformation. Many programming languages have been developed to make it easy to define meaningful lenses under specific lens laws [4, 9]. They basically give a solution by either permitting to use limited primitives and their combinations or imposing a strong syntactic restriction to write bidirectional programs. If general-purpose languages are used for bidirectional programming, the conformance to the desirable lens laws should be checked for each program. The problem of checking the conformance is, however, in general undecidable because it involves a kind of functional equalities. This is why many bidirectional programming languages have been proposed, where specific lens laws always hold due to a careful design of the languages.

Fischer et al. [3] have shown that weaker lens laws can imply some of the core lens laws which are useful to design bidirectional programming languages. They give a ‘clear picture’ of lens laws where relationship over 9 lens laws shown in Fig. 1 and Fig. 2 except two, (WeakPutGet) and (Undoability), is investigated to show which combination of weaker laws can imply a core law. Implications among lens laws often help to find their unexpected interaction and give a clear insight to bidirectional transformation. For example, every bijective lens (that satisfies the (StrongGetPut) and (PutPut) laws) is found to be very-well-behaved (that
11 lens laws to identify an essence of bidirectional transformation.

is, to satisfy the (GetPut), (PutGet) and (PutPut) laws from the facts that the (PutGet) law implies (PutInjectivity) and the conjunction of the (StrongGetPut) and (PutInjectivity) laws implies (PutPut). Fischer et al. introduced several implications to show that a well-behaved lens can be uniquely obtained only from a put function as long as put satisfies the (PutSubjectivity), (PutTwice) and (PutInjectivity) laws.

A major goal of the present report is to improve Fischer et al.’s clear picture of lens laws. Specifically, we add more two lens laws, (WeakPutGet) and (Undoability), which have been introduced for a practical use [2, 6, 7] and find all implications among the 11 lens laws to identify an essence of bidirectional transformation. This report describes the following two contributions:

- Relationship among lens laws including the (WeakPutGet) and (Undoability) laws is investigated and the laws are shown to be classified based on it (Section 2).
- Implications among lens laws and their conjunctions are given as many as possible (Section 3). They are summarized by a complicated web structure shown in Fig. 4.

Note that the set of implications introduced in the present report is not shown to be complete in the sense that there may exist an implication among laws which can not be derived from the set. It is left as future work.

Related Work. As mentioned earlier, the present work is an improvement of a clear picture of lenses introduced by Fischer et al. [3]. They give only a few implications among lens laws except (WeakPutGet) and (Undoability). The present report covers much more implications some of which are not trivial.

Hidaka et al. [7] gives a classification to bidirectional transformation approaches including properties like lens laws required for well-behavedness. They just present the properties independently of each other and do not mention anything about their relationship.

Stevens [10] gives implications among a few of properties of symmetric lenses, in which sources and views are evenly treated and get takes two arguments like put. Some of the implications she presents hold also for asymmetric ones as shown in the present report. It would be interesting to consider a complete picture similar to ours for symmetric lens laws.

2 LENS LAWS AND THEIR CLASSIFICATION

We shall give a brief summary to the 11 lens laws in Fig. 1 and Fig. 2 and show implications among them, e.g., (StrongGetPut) implies (GetPut) and (PutPut) implies (PutTwice). Combining the implications tells us that all the lens laws are classified into three. Except the (WeakPutGet) law, every lens law is weaker than or equal to exactly one of the (StrongGetPut), (GetPut), or (PutPut) laws; the (WeakPutGet) law is strictly weaker than both the (StrongGetPut) and (PutPut) laws. Therefore we classify a set of lens laws into three families according to which of three laws, (StrongGetPut), (GetPut), and (PutPut), implies the law. We call the three families, GetPut, GetPut, and PutPut, respectively. The only (WeakPutGet) law can belong to two families.

In the rest of this report, we write sets of sources and views as S and V, respectively. We denote by $\mathcal{L}(S, V)$ for a set of all possible combinations of get and put functions, i.e., $\mathcal{L}(S, V) = \{S \to V\} \times \{S \times V \to S\}$. For demonstrating examples of lenses, we will use sets $\mathbb{Z}$, $\mathbb{N}$ and $\mathbb{Q}$ of integers, non-negative integers and rationals, respectively, and denote by $(x, y)$ for an element of $X \times Y$ with $x \in X$ and $y \in Y$. For $x \in \mathbb{Q}$, $\lfloor x \rfloor$ denotes the largest integer less than or equal to $x$. Most of the examples presented here may look elaborate and artificial so as not to satisfies as many other lens laws as possible.

2.1 GetPut Family

The GetPut family consists of six lens laws, (StrongGetPut), (GetPut), (Undoability), (WeakPutGet), (SourceSubjectivity), and (PutSubjectivity), all of which are entailment of the (StrongGetPut) law.

Law (StrongGetPut). This law indicates that the source is determined only by the view even though the view has less information than the source in general. If the view is given by get with a source, then the source is obtained by put independently of the original source. Under this law, the get function is left-invertible with the put function. For example where $S = V = \mathbb{Z}$, a pair of the get and put functions defined by $\text{get}(s) = 2s$ and $\text{put}(s, v) = \lfloor v/2 \rfloor$ satisfies the (StrongGetPut) law.

Law (GetPut). This law is literally a weakened version of the (StrongGetPut) law. Under this law, the source does not change as long as the view is the same as that obtained by the original source. For example where $S = V = \mathbb{Z}$, a pair of the get and put functions defined by $\text{get}(s) = 2s$ and $\text{put}(s, v) = v = s$ satisfies the (GetPut) law but not the (StrongGetPut) law.

Law (WeakPutGet). This law is literally a weakened version of the (PutGet) law. While the (PutGet) law requires the equality

$\forall s \in S, \exists v \in V, \quad \text{put}(s, \text{get}(\text{put}(s, v))) = \text{put}(s, v)$

(WeakPutGet)

$\forall s \in S, \forall v \in V, \quad \text{put}(\text{put}(s, v)) = \text{put}(s, v)$

(Undoability)

$\forall s \in S, \forall v \in V, \quad \text{put}(s, \text{get}(s)) = s$

(PutTwice)

$\forall s \in S, \exists v \in V, \quad \text{put}(s, v) = s$

(SourceSubjectivity)

$\forall s \in S, \exists v \in V, \quad \text{put}(s, v) = s$

(PutSubjectivity)

$\forall s, s' \in S, \forall v, v' \in V, \quad \text{put}(s, v) = \text{put}(s', v') \Rightarrow v = v'$

(PutInjectivity)

$\forall s, s' \in S, \forall v, v' \in V, \quad \text{put}(s, v) = \text{put}(s', v') \Rightarrow v = v'$

Figure 2: Other lens laws
Although it has been investigated in a few papers \[2, 5, 7\] the many lens laws are studied. Indeed, this law is the only exception and it is an immediate consequence of the (StrongGetPut) law.

**Law (Undoability).** This law implies that any source can be recovered with the view obtained from the source itself no matter how source is updated by a different view. For example where \( S = V = \mathbb{Z} \), a pair of the \( \text{get} \) and \( \text{put} \) functions defined by \( \text{get}(s) = \lfloor s/2 \rfloor \) and \( \text{put}(s, v) = 2v - s + 1 + 2 \lfloor s/2 \rfloor \) satisfies the (Undoability) law. Although it has been investigated in a few papers \([2, 5, 7]\), the (Undoability) law is not mentioned even by Fischer et al. \([3]\) where many lens laws are studied. Indeed, this law is the only exception in Fig. 2 that they do not explore. This is probably because it can be easily derived from the (GetPut) and (PutPut) laws. However, we think that the (Undoability) law is one of important lens laws because it is as powerful as the other strong lens laws by combining with weak lens laws as we will see later.

**Law (SourceStability).** This law requires every source is stable for a certain view. Defining the \( \text{get} \) function that returns the corresponding view for a given source, the pair conforms the (SourceStability) law. For example where \( S = V = \mathbb{Z} \), \( \text{put}(s, v) = (v-s+1)v \) satisfies the (PutTwice) law for which there are infinitary many choices of the \( \text{get} \) function to have the (GetPut) law.

**Law (PutSurjectivity).** This law requires literally surjectivity of the \( \text{put} \) function. This law is a weakened version of the (SourceStability) law. For example where \( S = V = \mathbb{Z} \), \( \text{put}(s, v) = 2s - 3v \) satisfies the (PutSurjectivity) law but not the (SourceStability) law.

The GetPut family makes an implication web as shown in Fig. 3(a) where a double arrow \( \Rightarrow \) stands for an implication between the two lens laws (e.g., \( \text{StrongGetPut} \Rightarrow \text{(GetPut)} \)) and a single arrow \( \rightarrow \) from the \( \wedge \) symbol stands for an implication from the conjunction of the two lens laws connected with \( \wedge \) to the lens law pointed by the arrow head (e.g., \( \text{(WeakPutGet) ∧ (SourceStability)} \Rightarrow \text{(GetPut)}. \))

Let us define six classes \( \text{StrongGetPut}, \text{GetPut}, \text{WeakPutGet}, \text{Undoability}, \text{SourceStability}, \) and \( \text{PutSurjectivity} \) of lenses as subsets of \( \mathcal{L}(S, V) \) corresponding six lens laws, e.g., \( \text{StrongGetPut} \triangleq \{(\text{get, put}) \in \mathcal{L}(S, V) \mid \forall s, s' \in S. \text{put}(s, \text{get}(s')) = s'\} \). Then every implication in the figure is shown in the following theorem by an inclusion among lens classes.

**Theorem 2.1.** The GetPut family has the following inclusions.

1. \( \text{StrongGetPut} \subseteq \text{GetPut} \subseteq \text{SourceStability} \subseteq \text{PutSurjectivity} \)

\( \text{StrongGetPut} \subseteq \text{GetPut} \subseteq \text{SourceStability} \subseteq \text{PutSurjectivity} \)

\(^3\)In \([2]\), a lens is said undoable when not only \( \text{Undoability} \) but also \( \text{(PutGet)} \) hold in our terminology.
get and put functions defined by $\text{get}(s) = \lfloor s/2 \rfloor$ and $\text{put}(s, v) = 2v$ satisfies the (PutGet) law.

**Law (ViewDetermination).** This law indicates that there is no distinct pair of views which generates the same source by the put function. Combining with the (SourceStability) law, it guarantees the uniqueness of the function to form a well-behaved lens [3]. For example where $S = V = Z$, $\text{put}(s, v) = 2^{|v|}(2v - 1)$ satisfies the (ViewDetermination) law.

**Law (PutInjectivity).** This law requires literally injectivity of the put function for each source fixed. This law guarantees that there is no distinct pair of views which leads the same source for the fixed original source. This law is a weakened version of the (ViewDetermination) law. The three law combination of (PutTwice), (PutSubjectivity) and (PutInjectivity) is equivalent to the two law combination of (SourceStability) and (ViewDetermination) [3]. For example where $S = V = Z$, $\text{put}(s, v) = 2^{|v|}v$ satisfies the (PutInjectivity) law but violates the (ViewDetermination) law.

The PutGet family makes an implication web as shown in Fig. 3(b). Let us define two classes PutGet, ViewDetermination, and PutInjectivity of lenses in a similar way to those of the GetPut family. Every implication in the figure is shown in the following theorem.

**Theorem 2.2.** The PutGet family has three inclusions.

1. PutGet $\subseteq$ WeakPutGet
2. PutGet $\subseteq$ ViewDetermination $\subseteq$ PutInjectivity
3. PutInjectivity $\cap$ WeakPutGet $\subseteq$ PutGet

**Proof.** Both inclusions, (1) and (2), are trivial. For (3), suppose that the (PutInjectivity) and (WeakPutGet) laws hold. Then we have $\text{put}(s, \text{get}(\text{put}(s, v))) = \text{put}(s, v)$ because of the (WeakPutGet) law. This equation implies $\text{get}(\text{put}(s, v)) = v$ by the (PutInjectivity) law, hence we have the (PutGet) law. $\square$

### 2.3 PutPut Family

The PutPut family consists of two lens laws, (PutPut) and (PutTwice), which forms a simple entailment of the (PutPut) law.

**Law (PutPut).** This law requires that the source obtained by repeatedly applying the put functions with many views is the same as that obtained by a single put application with the last view. It plays an important role for state-based lenses, that is, the history of updates can always be ignored. For example where $S = V = Z$, $\text{put}(s, v) = 2 \lfloor s/2 \rfloor - 2 \lfloor v/2 \rfloor + v$ satisfies the (PutPut) law.

**Law (PutTwice).** This law imposes ‘idempotency’ of the put function applied with the fixed view. This law is obviously a weakened version of the (PutPut) law. For example where $S = V = Z$, $\text{put}(s, v) = 2 \lfloor (s - v)/2 \rfloor + v$ satisfies the (PutTwice) law but violates the (PutPut) law.

The PutPut family makes a simple implication web as shown in Fig. 3(c). Let us define two classes PutPut and PutTwice of lenses in a similar way to those of the GetPut family. The implication in this family is shown in the following theorem whose proof is straightforward.

**Theorem 2.3.** The PutPut family has an inclusion.

\[ \text{PutPut} \subseteq \text{PutTwice} \]

### 3 Association Beyond Families

We have seen that a single lens law does not entail any lens law in the different family except for the case involving the (WeakPutGet) law. In this section, we investigate inclusions of the form $C_1 \cap C_2 \subseteq C$ beyond families. Specifically, possible inclusions of this form are presented where either (a) $C_1$ and $C$ belong to the same family or (b) $C_1$ and $C_2$ and $C$ belong to different families each other. All of those inclusions are proper although their proofs are omitted in the present report.

3.1 Equivalence under Another Law

First, possible implications of the form $C_1 \cap C_2 \subseteq C$ are studied where $C_1$ and $C$ belong to the same family and $C \subseteq C_1$. This type of inclusions indicates that $C_1$ and $C$ are equivalent within $C_2$, i.e., $C_1 \cap C_2 = C \cap C_2$.

In the GetPut family, an inclusion of this type is found.

**Theorem 3.1.** The following inclusion holds.

\[ \text{PutSurjectivity} \cap \text{PutTwice} \subseteq \text{SourceStability} \]

**Proof.** Suppose that the (PutSurjectivity) and (PutTwice) laws hold. For $s \in S$, (PutSurjectivity) gives $s' \in S$ and $v \in V$ such that $\text{put}(s', v) = s$. Then we have

\[
\begin{align*}
\text{put}(s, v) &= \text{put}(\text{put}(s', v), v) = s' \quad \text{by (PutTwice)} \\
&= \text{put}(s', v) \\
&= s \quad \text{by (GetPut)}
\end{align*}
\]

hence the (SourceStability) law holds taking $v$. $\square$

This inclusion gives an equivalence relation in the GetPut family under a lens law belonging to another family, that is,

**SourceStability $\cap$ PutTwice $\subseteq$ PutSurjectivity $\cap$ PutTwice**

The following theorem shows a inclusion where two lens classes in the GetPut family are involved as well as the above but those two are not related by inclusion. Nevertheless it leads their equivalence under another lens laws in a different family as we will see later.

**Theorem 3.2 ([2, 5]).** The following inclusions hold.

\[ \text{GetPut} \cap \text{PutPut} \subseteq \text{Undoability} \]

**Proof.** Suppose that the (GetPut) and (PutPut) laws hold. Then we have the (Undoability) law because

\[
\text{put}(s, v), \text{get}(s) = \text{put}(s, \text{get}(s)) \quad \text{by (PutPut)} \\
= s \quad \text{by (GetPut)}
\]

This theorem leads equivalence of the (GetPut) and (Undoability) laws under (PutPut) law as follows:

\[ \begin{align*}
\text{GetPut} \cap \text{PutPut} \\
&\subseteq \text{Undoability} \cap \text{PutPut} \quad \text{by Theorem 3.2} \\
&\subseteq \text{Undoability} \cap \text{PutSurjectivity} \cap \text{PutTwice} \cap \text{PutPut} \\
&\text{by Theorem 2.1(2) and Theorem 2.3} \\
&\subseteq \text{WeakPutGet} \cap \text{SourceStability} \cap \text{PutPut} \\
&\text{by Theorem 3.1 and Theorem 2.1(3)} \\
&\subseteq \text{GetPut} \cap \text{PutPut} \quad \text{by Theorem 2.1(4)}
\end{align*}
\]

which indicates $\text{GetPut} \cap \text{PutPut} = \text{Undoability} \cap \text{PutPut}$. 

K. Nakano
Towards a Complete Picture of Lens Laws

Figure 4: Implication among Lens Laws

In the PutGet family, three inclusions of the form $C_1 \cap C_2 \subseteq C$ are presented where $C_1$ and $C$ belong to the PutGet family and $C \subseteq C_1$.

**Theorem 3.3.** The following inclusions hold.

1. $\text{ViewDetermination} \cap \text{GetPut} \subseteq \text{PutGet}$
2. $\text{PutInjectivity} \cap \text{PutTwice} \subseteq \text{ViewDetermination}$

**Proof.** For (1), suppose that the (ViewDetermination) and (GetPut) laws hold. By the (GetPut) law, we have $\text{put}(\text{put}(s, v), \text{get}(\text{put}(s, v))) = \text{put}(s, v)$. Since this equation implies $\text{get}(\text{put}(s, v)) = v$ by the (ViewDetermination) law, we have the (PutGet) law.

For (2), suppose that the (PutInjectivity) and (PutTwice) laws hold. When $\text{put}(s, v) = \text{put}(s', v')$, we have

$$\text{put}(\text{put}(s, v), v) = \text{put}(s, v)$$

by (PutTwice)

$$= \text{put}(s', v')$$

by the assumption

$$= \text{put}(\text{put}(s', v'), v')$$

by (PutTwice)

$$= \text{put}(\text{put}(s, v), v')$$

by the assumption

This equation implies $v = v'$ by the (PutInjectivity) law, hence we have the (ViewDetermination) law.

These inclusions makes two or three laws in the PutGet family equivalent under another law in a different family:

$$\text{ViewDetermination} \cap \text{GetPut} = \text{PutGet} \cap \text{GetPut}$$

$$\text{PutInjectivity} \cap \text{PutTwice} = \text{ViewDetermination} \cap \text{PutTwice}$$

### 3.2 Implication of Combination

Next, possible implications of the form $C_1 \cap C_2 \subseteq C$ are studied where $C_1$, $C_2$, and $C$ belong to different families. Two inclusions of this type are found.

**Theorem 3.4.** The following inclusions hold.

1. $\text{SourceStability} \cap \text{ViewDetermination} \subseteq \text{PutTwice}$
2. $\text{StrongPutPut} \cap \text{PutInjectivity} \subseteq \text{PutPut}$

**Proof.** For (1), suppose that the (SourceStability) and (ViewDetermination) holds. By the (SourceStability) law, we take $v'$ such that $\text{put}(\text{put}(s, v), v') = \text{put}(s, v)$. This equation implies $v' = v$ by the (ViewDetermination) law. Then we have

$$\text{put}(\text{put}(s, v), v)$$

$$= \text{put}(\text{put}(s, v), v')$$

by $v = v'$

$$= \text{put}(s, v)$$

by $\text{put}(\text{put}(s, v), v') = \text{put}(s, v)$

which indicates the (PutTwice) law.

For (2), suppose that the (StrongGetPut) and (PutInjectivity) laws hold. By the (StrongGetPut) law, we have

$$\text{put}(\text{put}(s, v), \text{get}(\text{put}(\text{put}(s, v), v'))) = \text{put}(\text{put}(s, v), v').$$

By applying the (PutInjectivity) law to this equation, we have

$$\text{get}(\text{put}(\text{put}(s, v), v')) = v'.$$

Then the (PutPut) law holds because

$$\text{put}(\text{put}(s, v), v') = \text{put}(s, \text{get}(\text{put}(\text{put}(s, v), v')))$$

by (StrongGetPut)

$$= \text{put}(s, v')$$

by the equation. \qed

### 3.3 Summary of Implications

Combining all implication theorems shown in the present report, we have a big web structure among 11 lens laws as shown in Fig. 4. This figure tells not only implications but equalities among lens laws and their conjunctions.

For example, the equivalence relation shown in [3, Theorem 2],

$$(\text{SourceStability}) \cap (\text{ViewDetermination}) \Leftrightarrow (\text{PutInjectivity}) \cap (\text{PutTwice}) \cap (\text{PutInjectivity}),$$

can be concluded from this figure by checking that the conjunction of the (SourceStability) and (ViewDetermination) laws entails the (PutInjectivity), (PutTwice), and (PutGet) laws, and vice versa.

For another example, any lens satisfying the (WeakPutGet), (SourceStability), and (ViewDetermination) laws can be found to be well-behaved because the figure leads to the (GetPut) and (PutGet) laws from the three laws. This holds even when the (PutInjectivity) law instead of (ViewDetermination).
4 CONCLUDING REMARK

A precise relationship among lens laws has been presented. Eleven lens laws which have been introduced in the literature on bidirectional transformation are found to relate to each other, one implies another and a combination of two implies another. The implication graph which shows all the relationship might be helpful to check lens laws and certify properties for a given bidirectional transformation.

Our goal is to give a ‘complete picture’ of lens laws from which we can derive all possible implications of the form $C_1 \land \cdots \land C_n \rightarrow C$ with classes $C_1, \ldots, C_n$ and $C$ of lens laws. To achieve the goal, it would be shown that every implication of the form which cannot be obtained from the implication graph has a counterexample. The complete picture will help us to understand the essence of bidirectional transformation.

REFERENCES

[1] François Bancilhon and Nicolas Spyratos. 1981. Update Semantics of Relational Views. ACM Trans. Database Syst. 6, 4 (1981), 537–575.
[2] Zinovy Diskin. 2008. Algebraic Models for Bidirectional Model Synchronization. In Model Driven Engineering Languages and Systems, 11th International Conference, ModELEs 2008, Toulouse, France, September 28 - October 3, 2008. Proceedings. Springer, 21–36.
[3] Sebastian Fischer, Zhenjiang Hu, and Hugo Pacheco. 2015. A Clear Picture of Lens Laws - Functional Pearl. In Mathematics of Program Construction - 12th International Conference, MPC 2015, Königswinter, Germany, June 29 - July 1, 2015. Proceedings. Springer, 215–223.
[4] J Nathan Foster, Michael B. Greenwald, Jonathan T. Moore, Benjamin C. Pierce, and Alan Schmitt. 2007. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. ACM Trans. Program. Lang. Syst. 29, 3 (2007), 17.
[5] Nate Foster, Kazutaka Matsuda, and Janis Voigtländer. 2010. Three Complementary Approaches to Bidirectional Programming. In Generic and Indexed Programming - International Spring School, SSGIP 2010, Oxford, UK, March 22-26, 2010. Revised Lectures. Springer, 1–46.
[6] Soichiro Hidaka, Zhenjiang Hu, Kazuhiro Inaba, Hiroyuki Kato, Kazutaka Matsuda, and Keisuke Nakano. 2010. Bidirectionalizing graph transformations. In Proceeding of the 15th ACM SIGPLAN international conference on Functional programming, ICFP 2010, Baltimore, Maryland, USA, September 27-29, 2010. 205–216.
[7] Soichiro Hidaka, Massimo Tisi, Jordi Cabot, and Zhenjiang Hu. 2016. Feature-based classification of bidirectional transformation approaches. Software and System Modeling 15, 3 (2016), 907–928.
[8] Yasunori Ishihara, Hiroyuki Kato, Keisuke Nakano, Makoto Onizuka, and Yuya Sasaki. 2019. Toward BX-Based Architecture for Controlling and Sharing Distributed Data. In IEEE International Conference on Big Data and Smart Computing, BigComp 2019, Kyoto, Japan, February 27 - March 2, 2019. 1–5.
[9] Hsiang-Shang Ko, Tao Zan, and Zhenjiang Hu. 2016. BiGUL: a formally verified core language for putback-based bidirectional programming. In Proceedings of the 2016 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, PEPM 2016, St. Petersburg, FL, USA, January 20 - 22, 2016. 61–72.
[10] Perdita Stevens. 2012. Observations relating to the equivalences induced on model sets by bidirectional transformations. ECEASST 49 (2012).