TIDY: A PBE-based Framework Supporting Smart Transformations for Entity Consistency in PowerPoint

Shuguan Liu
liu_shuguan@126.com
State Key Lab for Novel Software Technology and
Department of Computer Science and Technology,
Nanjing University, Nanjing, China

Chang Xu
changxu@nju.edu.cn
State Key Lab for Novel Software Technology and
Department of Computer Science and Technology,
Nanjing University, Nanjing, China

ABSTRACT

Programming by Example (PBE) is increasingly assisting repetitive tasks, such as text editing and spreadsheet manipulation, yet existing work falls short on dealing with rich-formatted documents like PowerPoint (PPT) files. This paper presents TIDY, a PBE-based framework, to assist entity transformations for their layout and style consistency in PPT files, in a way adaptive to entity contexts. TIDY achieves this by smartly examining entities' transformation histories and selectively isolating them into relevant segments. Compared to existing work, TIDY alleviates the requirement on example inputs and realizes better fault tolerance, preferable for PPT files, which expect efficient, interactive, and reliable operations. We implemented TIDY as a prototype tool and integrated it into PowerPoint. Its experiments reported that TIDY generally supports 86.4% PPT operations, and suggests correct transformations with a F1-score of 72.1–88.0%. Besides, it took only marginal time overhead of several milliseconds, validating its effectiveness and usefulness.

CCS CONCEPTS

• Software and its engineering→ Programming by example;

KEYWORDS

Programming by Examples, Entity Transformation, PowerPoint

1 INTRODUCTION

Internetware applications are featured by context-awareness and smart adaptation. We consider programming by examples (PBE) techniques are also enabling ones for Internetware applications, as they learn from contexts (users' inputs) and make required adaptations to replace human repetitive actions (automating task executions by synthesized actions). PBE [1] is a sub-field of program synthesis [2], can free people with no programming background from those troublesome tasks. Given some input-output examples as specification, a PBE technique can synthesize a program that satisfies the specification and generalizes well to new inputs. PBE has been widely applied in many application domains such as data wrangling [3, 4] and code transformations [5]. And it has already been used to improve the efficiency of people using Internetware applications [6].

However, PBE also has its limitations. Most PBE systems require users to enter a special mode to provide examples, which interrupts users' workflow and actually increases users' workload. In the domain of document editing, Miltner et al. proposed modeless synthesis, which is of referential significance to us. Their system identifies related examples by observing users silently as they make changes to the document and suggests transformations that can apply at other locations [7]. Besides, although the specifications consisting of examples are much easier for users to provide, it may lead to ambiguity, because there may be many programs that satisfy the examples. To get an intended program, we may need several examples especially for PowerPoint, an expressive application with rich formats. And this is a bit unbearable because unlike spreadsheet having dozens or even hundreds of cells that need to complete the same task, in a PowerPoint page, there may be only several entities need to do so. And the more examples needed to get an intended program, the fewer entities that can actually benefit from the program.

For the above problems, this paper proposes the following solution: we don't need users to provide examples in a special mode, instead, users' intentions (tasks to be completed) are identified automatically through previous operations and context; then we will give some suggestions of recommended operations to let the relevant entities to complete the same task. Rather than pursuing the only intention, we might suggest operations of several possible tasks and then find out the exact task that the user wants to accomplish according to the user's feedback.

This paper makes the following contributions: a domain-agnostic framework providing recommendations for subsequent entities and operations; and its specific technical implementation in the domain of PowerPoint. In addition, our work has other advantages. First, multi-step framework (TIDY$^{\beta}$) can deal with multiple steps (a step is an operation set on one entity) in noise-free operation history and identify the intention they may share. It is upgraded version of single-step framework (TIDY$^{\alpha}$) whose input is one step. Second, we added two extra functions, undo and redo. For some people may not have a clear intention when making PowerPoint pages, these two functions can help them try different ideas easily.

We developed a PowerPoint add-in based on single-step framework which can identify two types of users' intentions. We evaluated its supportability for common operations and its support effects for two output types of single-step framework. The evalua-

tion results are satisfactory: 86.4% of common operations can be supported by the add-in; the average F1-score of two output types was 88.01% and 72.14% respectively, and the average calculating time was 6.8ms and 8.6ms respectively.

The rest of this paper is organized as follows. Section 2 and Section 3 present single-step and multi-step framework, respectively. Section 4 introduces the single-step framework implementation and its evaluation is conducted in Section 5. Section 6 discusses related work and Section 7 concludes the paper.

2 TIDY$^{\alpha}$: SINGLE-STEP FRAMEWORK

This section presents the basic version of the framework, single-step framework. It only considers the step just performed, so the types of intentions can be identified by it are comparatively limited. An overview of the framework is described in Section 2.1, followed by definition of some concepts in Section 2.2. The five-phase process is introduced in Section 2.3 through Section 2.7.

2.1 Overview

Single-step framework takes the step just performed as the input and provides suggestions to assist following transformations. It is a little unconvincing to interpret a step as a complicated intention. Since most people tend to have neat and orderly PPT pages, a more reasonable aim state is the layout and style of some entities will tend to be consistent and the step just performed can be interpreted as the first move to achieve the aim state. According to the entity context, the intention of consistency can be refined into two more specific types of intentions.

The first one is to take the input entity as the target and let relevant entities to aligned with it on the input-changed attribute(s). In Figure 1 (a), there are five entities labeled from e_0 to e_4. The input is changing left (the distance between entity and the left edge of the page) of e_0 to 8, so the framework predicts the left of e_1 and e_2 will also be set to 8 and gives corresponding recommended operations as the output (Figure 1 (b)). We call the output of this type of intentions type 1 output.

The second one happens if some entities are highly relevant and the input can be interpreted as the input entity is approaching this highly relevant entity set. In this situation, the target might be the input entity or that highly relevant entity set. We call the output of this type of intentions type 2 output. In Figure 2 (a), the input can be interpreted as e_0 is approaching e_1 and e_2, so we predict the input entity and this highly relevant entity set will become closer to each other in other attributes like width and fore color (Figure 2 (b)). In type 2 output, entities in the highly relevant entity set also should be relevant to the input entity, or it would be too tough for them to get close to each other.

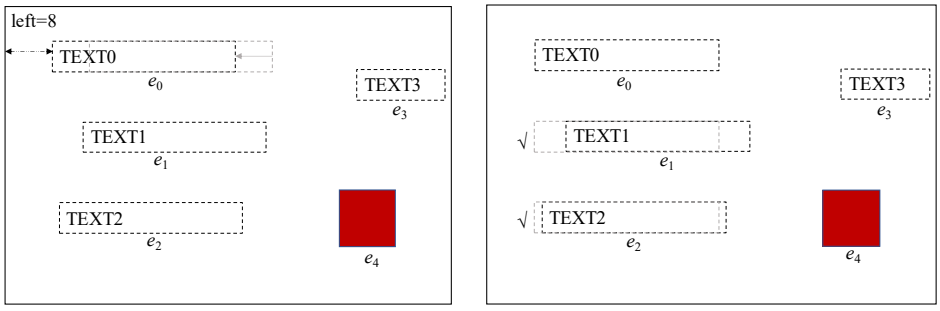
After getting an input, the five-phase process starts immediately. The main idea is to get those entities which are relevant to the input entity as candidate output entities, and then obtain the output based on the output type. Otherwise, we add entity selection and attribute selection to reduce the workload of calculating entity relevance.

Entity selection. Whatever the output type is, output entities need to be relevant to the input entity. So before calculating entity relevance formally, some entities can be excluded for they have obvious differences from the input entity. In Figure 1 (a), the input entity is e_0, a text box. After entity selection, rectangular shape e_4 is excluded and e_1, e_2 and e_3 are reserved as candidate entities.

Attribute selection. It will take too much time if we bring all attributes in entity relevance calculation. In addition, attributes are not equally important so they should not be regarded equally in next phase. Therefore, in this phase, we will measure the importance of attributes and exclude useless attributes. In Figure 1 (a), attributes such like shadow or dash style will be excluded since their values are uniform within all candidate entities. Every selected attribute will get an importance score.

Calculating entity relevance. In the third phase, we pick candidate output entities according to the relevance between each candidate entity and the input entity. Input-changed attributes should have their special treatment because both their initial values and current values of the input entity matter while for other attributes, only current values matter. For this reason, this phase consists of two stages. The first stage calculates relevance using selected attributes. The second stage updates the results of the first stage using input-changed attributes.

Judgment of output type. We judge the output type by whether we can find a highly relevant entity set that meets the conditions. In Figure 1 (a), we can't find a highly relevant entity set like e_1 and e_2 in Figure 2 (a), so its output type is type 1. If there are multiple highly relevant entity sets, there will be multiple mutually



(a) **(b)**

Figure 1: (a) input (<e_0, left, 8>) (b) output (<e_1, left, 8 >, <e_2, left, 8>)

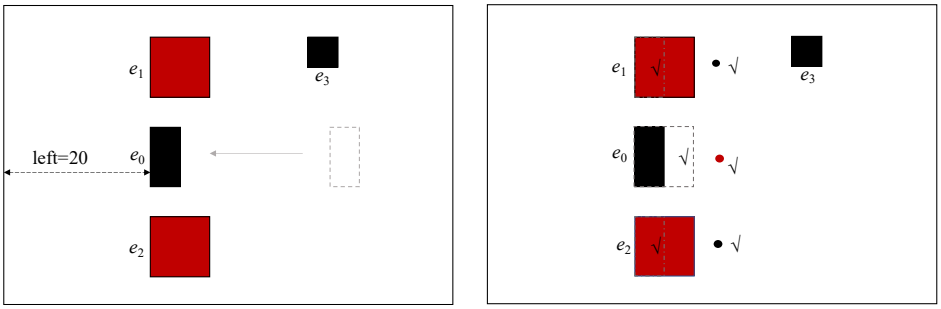


Figure 2: (a) input (<e_0, left, 20>) (b) output (<e_0, width, 10>, <e_0, color, red >, <e_1, width, 5>, <e_1, color, black>, <e_2, width, 5>, <e_2, color, black)>)

exclusive outputs. We devise an algorithm to search for highly relevant entity set(s).

Obtaining output. In the fifth phase, we obtain the output according to the result of last phase. After getting the output, we provide recommended operations to the user immediately. In the PowerPoint add-in we developed, users only need to click √ buttons to implement recommended operations.

2.2 Definition

An entity e = (*attribute*$_1$: *value*$_1$, *attribute*$_2$: *value*$_2$, …, *attribute*$_m$: *value*$_m$) is a set of key-value pairs consisting of attributes and corresponding attribute values. In PowerPoint, an entity can be a picture, a textbox or a rectangle shape, etc. Different entities may exist different attributes. For example, a rectangle shape without text doesn't exists attributes related to font, in other words, its values of font-related attributes are all null. Every entity exists attribute id and labeled by the id value uniquely.

An operation *op* = < *id*, *attribute*, *value*> is a record of user's actual transformation, the *attribute* of an entity labeled *id* is set as *value*. The *attribute* must be one of the read-write attributes the entity labeled *id* exists and the *value* can't be null. Some actual transformations which may cause additions or deletions of attributes or entities are out of the scope of this paper for they can't be recorded by our defined operation.

The input of single-step framework *input* = (*op*$_1$, *op*$_2$, …, *op*$_n$) = (<*id*, *attribute*$_1$, *value*$_1$>, <*id*, *attribute*$_2$, *value*$_2$>, …, <*id*, *attribute*$_n$, *value*$_n$>) is an operation set that the operations in it correspond to the same entity and distinct attributes. Only the last transformation will be recorded if there are more transformations than one corresponding to the same attribute. Input entity is the entity labeled *id*. Input-changed attributes *input-changed attribute set* = (*attribute*$_1$, *attribute*$_2$, … *attribute*$_n$) are all attributes whose values are changed by the input.

The output of single-step framework *output* = (*op*$_1$, *op*$_2$, …, *op*$_k$) is also an operation set. If the output type is type 2, there are some mutually exclusive relationships between the operations.

Other useful information is stored in *context*, e.g., current values of all entities in their existing attributes, initial values of the input entity in input-changed attributes and calculation results from previous phases.

2.3 Entity Selection

In this phase, we exclude some entities having obvious differences from the input entity and get candidate entities. The most important difference between entities is the difference in entity categories, e.g., the difference between a picture and a rectangle shape is bigger than the difference between two rectangle shapes with many distinct attribute values. We select entities that belong to the same category as the input entity to be candidate entities.

2.4 Attribute Selection

Before calculating entity relevance, we need to exclude some useless attributes and give every selected attribute an importance score as weight in calculating entity relevance. The importance of an attribute can be measured from three ways.

Firstly, the importance of an attribute is affected by its own value distribution. We exclude attributes whose values are uniform within all candidate entities.

Secondly, the importance of an attribute is affected by the input entity. To ensure each selected attribute has a reference value when calculating entity relevance, we exclude those attributes that the input entity doesn't exist.

Thirdly, the importance of an attribute is affected by input-changed attributes. We tend to give higher importance score to the attribute connected closely with input-changed attributes. As for how to measure the connection between two attributes, it's determined by the specific application. For example, in PowerPoint whose object model is a tree in which each leaf node is an attribute (Figure 3), we can measure the connection between two attributes by the reciprocal of their shortest path length.

2.5 Calculating Entity Relevance

To start with, we use selected attributes to calculate relevance between each candidate entity and the input entity. Then we use input-changed attributes to update relevance and pick candidate output entities.

Calculation using selected attributes. Attributes can be divided into two main categories according to their data types: numeric

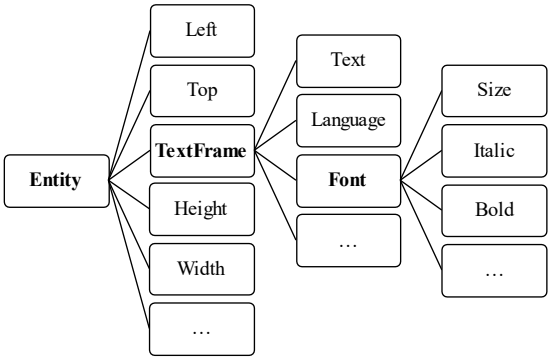


Figure 3: Object model of PowerPoint

attributes and tag attributes. Numeric attribute values can be added or subtracted while tag attributes can only compare whether their values are equal. When calculating entity relevance using selected attributes, the two categories of attributes should be calculated separately, and then we combine two results.

The relevance of an arbitrary entity e_j and the input entity using numeric selected attributes $relevance_{1j}$ is defined as follows.

$$relevance_{1j}=e^{\wedge}(-\sqrt{\sum_{i=1}^{count_1}\frac{score_i\cdot(value_{ij}-value_{i0})^2}{score\cdot s_j^2}})$$

In the above expression, *score* is the average importance score of all numeric selected attributes while $score_i$ is the importance score of $attribute_i$. $value_{ij}$ and $value_{i0}$ denote attribute value of e_j and the input entity in $attribute_i$, respectively. $count_1$ is amount of numeric selected attributes and s_j is the standard deviation of $attribute_i$ values in all candidate entities.

The relevance of an arbitrary entity e_j and the input entity using tag selected attributes $relevance_{2j}$ is defined in the below expression in which $count_2$ is the amount of tag selected attributes.

$$relevance_{2j}=\frac{\sum_{i=count_1+1}^{count_1+1+count_2}(B_{ij}\cdot score_i)}{\sum_{i=count_1+1}^{count_1+1+count_2}score_i}\qquad B_j=\begin{cases}1, value_{ij}=value_{i0}\\0, value_{ij}\neq value_{i0}\end{cases}$$

Then we define the relevance of an arbitrary entity e_j and the input entity $relevance_j$ using all selected attributes as follows.

$$relevance_j=\frac{relevance_{1j}\cdot\sum_{i=1}^{count_1}score_i+relevance_{2j}\cdot\sum_{i=count_1+1}^{count_1+1+count_2}score_i}{\sum_{i=1}^{count_1}score_i+\sum_{i=count_1+1}^{count_1+1+count_2}score_i}$$

Calculation using input-changed attributes. For every input-changed attribute, if the current value of an arbitrary entity e_j is equal to the initial value or current value of the input entity, then we increase the $relevance_j$. Then we set a relevance threshold to select candidate output entities. If the relevance between an entity and the input entity is higher than the threshold, the entity becomes a candidate output entity.

2.6 Judgment of Output Type

If we can find a highly relevant entity set among candidate output entities, the output type will be determined as type 2. A highly relevant entity set should satisfy the following conditions:

(1) There is at least one read-write attribute whose current values of all entities in the entity set are uniform but are not equal to the current value of the input entity.

(2) For the tag attribute, if the attribute values within the union of {the input entity} and the entity set are uniform before input, they should not be not uniform after input; for the numeric attribute, the variance of the attribute values within the union of {the input entity} and the entity set should not get bigger after input.

(3) The relevance within the entity set should be higher than the relevance between the entity set and the input entity.

There may be multiple highly entity sets that meet three conditions above and each entity set corresponds to a type 2 output, so we won't stop after having one highly entity set. Considering that the number of type 2 outputs that can be displayed in the user interface is limited, it is not necessary to find out all highly relevant entity sets, and the calculation can be stopped as long as the upper limit (upper_limit, a predefined constant) is reached. The algorithm for judging the output type and searching such entity set(s) is presented in Algorithm 1.

We start with the easiest condition, $condition_1$. We divide candidate output entity set into several id sets according to values then put entity sets that meet $condition_1$ into Q with the attribute (Line 2-5). Then we continuously merge two members in Q if they have the same entity set (Line 6), e.g. $<id\_set_x, attr\_set_y>$, $<id\_set_x, attr\_set_z> \rightarrow <id\_set_x, attr\_set_y \cup attr\_set_z>$. If an entity set in Q (it still meets $condition_1$) also meets $condition_2$ and $condition_3$, the entity set (with its bound attribute set) becomes a type 2 output (Line 11-14). Or the entity set will be split into complementary sets of each member, e.g. $(id_A, id_B, id_C) \rightarrow (id_A, id_B), (id_A, id_C), (id_B, id_C)$. These new generated entity sets (still meets $condition_1$) will be added into Q in the proper place (Line 15-19). If

Algorithm 1: Judgment of output type

Global variables: *context*, *type2_output_list*
Output: *output_type*

```
1:  l=limt; cnt=0; Q.Clear(); type2_output_list.Clear();
2:  for each attr_i in selected attribute set U input-changed attribute set
3:      candidate output entity set→id_set_i1, id_set_i2, … , id_set_in;
4:      Q.Add(<id_set_ij,(attr_i)>) if |id_set_ij|>=2 and input id∉id_set_ij;
5:  end for
6:  Q.Merge(id_set, =);
7:  Q.Sort(|id_set|*|attr_set|, desc);
8:  While Q.empty==false &&l>0
9:      _id_set, _attr_set=Q.first.id_set, Q.first.attr_set;
10:     Q.Delete_first();
11:     if _id_set meets condition_2 &&_id_set meets condition_3
12:         type2_output_list.Add(<_id_set,_attr_set>);
13:         l--; cnt++; continue;
14:     end if
15:     if |_id_set|>=3
16:         for each id in _id_set
17:             insert/merge _id_set-(id) into Q with _attr_set;
18:         end for
19:     end if
20: if cnt==0 output_type=1;
21: else output_type=2;
22: end if
23: return output_type;
```

there is at least one type 2 output, the output type is 2 (Line 20-23).

2.7 Obtaining Output

The methods of two output types to obtain the output are slightly different. But their main idea is the same since the output is an operation (<*id*, *attribute*, *value*>) set. To begin with, we get the ranges of entities, then the ranges of attributes, finally we get the value according its corresponding entity and attribute.

The ranges of entities in type 1 output are from candidate output entities got from the third phase. We may raise the relevance threshold for a stricter selection. As for type 2, there is probably more than one output. For each entity set obtained from the former phase, the ranges of entities in its corresponding output are all entities in this highly relevant entity set.

Then it comes to the ranges of attributes. In type 1 output, the output attribute set equals to the input-changed attribute set for most of output entities. But sometimes the former is a subset of the latter, e.g., an output entity already has the same value as the input entity in an input-changed attribute and that attribute won't be in the output entity's output attribute set. In type 2 output, every output entity's output attribute set is the attribute set bound to its entity set in the former phase.

In type 1 output, for each <*id*, *attribute*> obtained above, the operation's value equals to the current value of the input entity in *attribute*. In type 2 output, the input entity and the highly relevant entity set aligned with each other, or in other words, the operation's value of one side equals to the current value of the other side in the same attribute.

Although both output types are a set of operations, output type 2's operations in different recommended directions for the same attribute are mutually exclusive, different from type 1 output whose all recommended operations are compatible. For example, in Figure 2(b), you cannot choose both the recommended operation of narrowing e_0 and the recommended operation of widening e_1. And if there are multiple type 2 outputs, they are also mutually exclusive.

3 TIDY$^{\beta}$: MULTI-STEP FRAMEWORK

This section presents the upgraded version of the framework, multi-step framework. It can deal with multiple steps in noise-free operation history, so it can identify more diverse and complicated intentions. Some new concepts are defined in Section 3.1 and we discuss how to get qualified inputs in Section 3.2. Overview is not provided because the overall workflow of the other phases is similar to the single-step framework. But each phase is upgraded for adapting multiple steps, which is presented from Section 3.3 to Section 3.6 respectively.

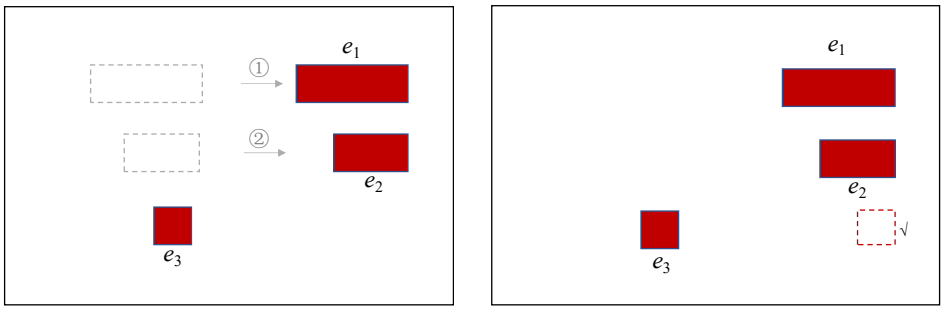
3.1 Definitions

Entities, operations, and context are defined as they are in the single-step framework. A step has the same definition as the input of single-step framework. Noise-free operation history is recorded in step-record sr = [$step_1$, $step_2$, …, $step_n$], a sequence of all steps in chronological order. Noise-free means that if an intention is to be completed, it will be done in successive steps, instead of having unrelated steps in the middle. A multi-step m= [$step_i$, $step_{i+1}$, …, $step_{j-1}$, $step_j$] is a subsequence of step-record.

A goal is a model to describe user's intention of multi-step and it doesn't have a fixed way of expression. In Figure 4 (a), e_1 and e_2 are moved to the right successively. We suppose that the user's intention is to make e_1 and e_2 right aligned, or in other words, unify the sum of left and width values. So, the goal of Figure 4 can be briefly expressed as <left, width, +> and its corresponding multi-step is [(<e_1, left,35>), (<e_2, left,40>)]. An operation or a step can also be interpreted. We use a candidate goal to describe user's intention of an operation or a step. For example, the first operation (<e_1, left,35>) shown in Figure 4 (a) has candidate goals such as right aligned, left aligned and center aligned. Candidate goals don't have big difference from goals, they're just not for multi-step.

The input of multi-step framework *input'* = {goal, m} consists of a goal and a multi-step. In Figure 4 (a), the input is {<left, width, +>, [(<e_1, left,35>), (<e_2, left,40>)]}. We define the input entity set as those entities who have completed the goal, in Figure 4, they are e_1 and e_2. And we define the input attribute set as those goal-related attributes, in Figure 4, they are left and width.

The output of multi-step framework has two forms, *output'*= (op_1, op_2, …, op_m), *output''*= [op_1, op_2, …, op_n]. It can be an oper-



(a) **(b)**

Figure 4: (a) (<e_1, left,35>), (<e_2, left,40>) (b) output (<e_3, left, 45>)

ation set like single-step framework and we show all recommended operations together. It can also be an operation sequence and we show the recommended operations one by one.

3.2 Obtaining Input(s)

After a step is submitted, unlike single-step framework whose input can be obtained directly, multi-step framework needs to take some effort to get the input because it involves identifying the goal from step-record. And there may be multiple inputs because there may be more than one identified goal. Three stages are required to get the input(s). To start with, we need to obtain candidate inputs. Then we make these candidate inputs complete and select qualified ones from them. The judgement of the output type is removed for the user's intentions are identified in this phase.

Obtaining candidate inputs. An operation can be interpreted as different user's intentions, in other words, an operation has a set of candidate goals. A step consists of one or more operations. The candidate goals of a step are equal to the union of the respective candidate goals of each operation contained in this step. In Table 1, step-record consists of 5 steps shown in list headers, and all candidate goals can be interpreted by these steps are shown in row headers. The cell corresponding to $step_i$ and $goal_j$ has ♦, which means $goal_j$ is one of candidate goals of $step_i$. For example, candidate goals of $step_5$ = ($goal_A$, $goal_B$, $goal_D$).

If successive steps contain the same candidate goal, then these steps are linked into a multi-step (colored in Table 1) with a shared goal. If a multi-step (as long as possible) has a shared goal and its last step is the latest step, then this multi-step and its shared goal can be a candidate input. The latest step must be included because the framework tries to obtain inputs immediately after a new step is submitted, and if the latest step is not included, then the multi-step should have been handled before. In Table 1, we obtain 3 candidate inputs: {$goal_A$, [$step_3$, $step_4$, $step_5$]}, {$goal_B$, [$step_4$, $step_5$]}, {$goal_D$, [$step_4$, $step_5$]}.

It is also important to note that the goal (a model) is not static. The parameters in the goal are determined by the steps it interprets. Both $goal_A$ and $goal_E$ may be right-aligned, but $step_3$ cannot be accepted by $goal_E$ (instantiated by $step_1$ and $step_2$), so $goal_A$ (instantiated by $step_3$) is added.

Making candidate inputs complete. When we obtaining candidate inputs, we only consider steps and their candidate goals. So, the input entity set, at this moment, only includes those entities in multi-step. However, some entities (don't have operations) may have completed the goal of the input when they were created, and by definition they should also be included by the input entity set. At this stage, we scale all other entities to figure out whether they satisfy the goal of each candidate input, and if so, we add those entities into the input entity set of corresponding candidate input.

Input selection. Some candidate inputs may be unconvincing because they don't have enough entities in their input entity sets. Different goals have different predefined minimum requirement for the number of elements in the input entity set. For example, left aligned and right aligned may need at least 2 entities in the input entity set while center aligned may need at least 3 entities in input entity set. Qualified candidate inputs become actual inputs. If there are multiple inputs, we treat them separately in the following phases.

Table 1: Step-record and its candidate goals.

	$step_1$	$step_2$	$step_3$	$step_4$	$step_5$
$goal_A$			♦	♦	♦
$goal_B$				♦	♦
$goal_C$		♦	♦	♦	
$goal_D$				♦	♦
$goal_E$	♦	♦			

3.3 Entity Selection

Some goals may be only for one or a few entity categories. In this phase, we retain entities of that or those entity categories as candidate entities. If there is no such clear requirement for entity category, all entities except attributes in the input entity set become candidate entities.

For some goals, only attributes in the input attribute set matter, e.g., the user intends to move every entity on the left half of the page to a symmetrical position on the right half. For the input with these goals, we can skip attribute selection and the first stage of calculating entity relevance because they focus on attributes except those attributes in the input attribute set, and we can go straight to the second stage of calculating entity relevance.

3.4 Attribute Selection

This phase share similar methods with attribute selection in single-step framework. We regard the input attribute set as the input-changed attribute set. And instead of deleting those attributes that doesn't exist for the input entity, we delete those attributes that doesn't exist for each entity in the input entity set. In addition, to get an importance score of every selected attribute, we also need to get a benchmark value interval ([min value in the input entity set, max value in the input entity set]) of each selected numeric attribute and a benchmark value set (values of entities in the input entity set) of each selected tag attribute.

3.5 Calculating Entity Relevance

The basic idea of this phase is roughly the same as the corresponding phase in the one-step framework. But the former approach is a generalization of the latter approach.

Calculation using selected attributes. In multi-step framework, the relevance of an arbitrary entity e_j and the input entity using numeric selected attributes $relevance_{1j}$ is defined as follows.

$$relevance_{1j} = e^{\wedge}\left(-\sqrt{\sum_{i=1}^{counti} \frac{score_i \cdot (\Delta value_{ij})^2}{score \cdot s_j^2}}\right)$$

$$\Delta value_{ij} = \begin{cases} 0, & min\_value_i <= value_{ij} <= max\_value_i \\ min\_value_i - value_{ij}, & value_{ij} < min\_value_i \\ value_{ij} - max\_value_i, & value_{ij} > max\_value_i \end{cases}$$

The min and max value of $attribute_i$ in the input entity set are represented as min_value_i and max_value_i, respectively. The other

variables have the same meaning as they do in the single-step framework.

The relevance of an arbitrary entity e_j and the input entity using tag selected attributes $relevance_{2j}$ is defined as follows. $value\_set_i$ is the benchmark value set consists of values of $attribute_i$ in the input entity set.

$$relevance_{2j} = \frac{\sum_{i=count_1+1}^{count_1+1+count_2} (B_j \cdot score_i)}{\sum_{i=count_1+1}^{count_1+1+count_2} score_i} \qquad B_{ij} = \begin{cases} 1, value_{ij} \in value\_set_i \\ 0, value_{ij} \notin value\_set_i \end{cases}$$

The expression of $relevance_j$ is the same as it in single-step framework.

Calculation using input attribute set. According to the goal of the input, there are two main types of treatment. The first type of treatment is similar to the approach in single-step framework. We update some entities' relevance got in the first stage of this phase, then we choose those entities with high relevance to be output entities. How to update relevance depends on the specific goal. For example, an input's goal is to turn the graphics block purple and original colors of entities in the input entity set are yellow or blue, then the relevance of all yellow or blue candidate entities should be increased in this stage. Another example, in the goal where the graphic blocks are arranged equidistantly (left values in the input entity set:115, 120, 125), the relevance of a candidate entity that is close to the target value (130) should be increased. The second type of treatment is used in the cases if we jump directly to this stage after entity selection. We exclude some candidate entities according to values of attributes in the input attribute set and get output entities. For example, if the goal is moving a numeric attribute value from interval A to interval B, then candidate entities whose value is not within interval A are directly excluded.

3.6 Obtaining Outputs

The method of obtaining the output is determined by specific goal. Just like type 2 output in TIDY$^{\alpha}$, the relationship between different outputs of the same input is mutually exclusive. And for an operation, although it may have multiple candidate goals, it has only one exact interpretation. So, if the goals of two inputs contain different interpretations of the same operation, then the outputs produced by the two inputs are also mutually exclusive.

4 IMPLEMENTATION

The single-step framework is implemented using C# to create a VSTO (Visual Studio Tools for Office) add-in for Microsoft Office PowerPoint.

In attribute selection, we select 20% of attributes (except the input-changed attributes) to be selected attribute set. These attributes determine 80% of relevance score in relevance calculation according to the Pareto Principle, and the other 20% of relevance score is determined by the input-changed attributes. We set the threshold of relevance score also as 0.2 to determine whether an entity can be candidate output entity. In the judgement of the output type, we set *upper_limit*=5. We compare the relevance within the entity set and the relevance between the entity set and the input entity by comparing the variance of numeric attribute values and the consistency of tag attribute values. In the last phase (obtaining outputs), if the output type is type 1, we raise the threshold of relevance score to 0.8 to get output entities.

In addition, we add two extra functions in the custom task pane, undo and redo, to undo and redo recommended operations.

5 EVALUATION

We conducted studies to evaluate the tool about its quality and efficiency. We raised following research questions.

RQ1 (Supportable range): *Does the tool support most of PPT operations?*

RQ2 (Effectiveness): *How accurate are the suggestions provided by the tool?*

RQ3 (Overhead): *How fast can the tool produce suggestions?*

5.1 Experimental Design and Setup

5.1.1 Study on RQ1

If an operation (whose attribute is $attribute_x$) can be part of the input, and another operation (whose attribute is also $attribute_x$) can be part of the output and can be done by the tool, then we determine that $attribute_x$ can be supported by the tool, and in turn we determine that operations whose attributes are $attribute_x$ can be supported by the tool. We selected 59 common read-write attributes from PowerPoint official documents. Each attribute used a sample to confirm whether it could be supported. This study didn't require participants.

Then we selected 15 most common attributes for the following studies from the attributes that could be supported.

5.1.2 Studies on RQ2 and RQ3

To evaluate the tool's support effect, we conducted two studies corresponding to two output types in single-step framework respectively. Both studies need participants and have requirements to them.

Participants. We recruited three participants composed of a teacher, a student and a public official. They have basic experience in using PowerPoint, but they are not PowerPoint professional designers and have no experience in developing PowerPoint add-ins. Compared with the dramatic and imaginative slides, they are more accustomed to making orderly and neat slides which is consistent with the idea of our framework.

Setup. Each sample was a PowerPoint page and a modifiable attribute set displayed in the note of the page.

In the study on type 1 output, first of all, we made the modifiable attribute set contain only one attribute. For the 15 attributes obtained in the first study, we designed 3 samples for each attribute with it as the only modifiable attribute. Entity quantity on the page was 5/10/15 respectively. Then, we made the modifiable attribute set contain multiple attributes. We selected 2, 3 and 4 attributes randomly from the 15 attributes (5 times for each attribute quantity) and designed total 15 samples with the set of selected

attributes as the modifiable attribute set. Entity quantity on every page was 10. Finally, we divided 60 samples into three and assigned 20 samples to every participant.

In the study on type 2 output, we designed samples with different numbers of attributes that satisfy conditon$_1$ (Section 2.6). We selected 1, 2, 3 and 4 attributes (6 times for each attribute quantity) randomly from the 15 attributes and designed total 24 samples. For each sample's page, there were several entity collections. Each entity collection had uniform values (different from the values of the entities outside the collection) on the selected attributes. Participates wouldn't be directly told about the selected attributes of a sample as the modifiable attribute set of every sample contained all the 15 attributes. We divided 24 samples into three and assigned 8 samples to every participant.

Requirements. In both studies, participants can only modify the values of the modifiable attribute set. Besides, they need to meet requirements as follows.

In the study on type 1 output, participants should find a reference entity whose values of all attributes within the modifiable attribute set will be modified in the very first and the value of other entities could only follow the reference entity if they are modified subsequently.

In the study on type 2 output, firstly, participants should choose an attribute set *y attribute set* (|*y attribute set*|>=1), and then choose an entity set *n entity set* (|*n entity set*|>= 2) that can be considered as a whole and an entity *m*. At the very beginning, the values of *n entity set* on every attribute in *y attribute set* should be uniform but not equal to the value of *m* and after completing all operations on this page, they should be all uniform. Secondly, take *m* as the first entity to be operated. There's no need to set every value of *m* to its target state at once. If all the reference values exist in the m or n entity set at a certain moment, uncheck *m* immediately, and then operate *n entity set* or *m* again. Thirdly, each entity in *n entity set* needs to be checked in turn after completing all operations on the entity *m*.

Process. To begin with, we recorded the participants' operations and divided them into the input and *actual operation set* (other operations). If the same entity had multiple operations with the same attribute, only the latest one would be recorded. In the study on type 1 output, the input is the operations of the first entity. In the study on type 2 output, the input is the operations of the first entity before it its first uncheck.

Afterwards, we handed over the input to the tool and obtained the output(s) as *predicted operation set*. In the study on type 1 output, to evaluate type 1 better, the tool skip judgment of output type and disabled type 2 output. In the study on type 2 output, we might get more than one *predicted operation set*.

Evaluation Measures. We measured calculating time and F1-score of each sample.

Calculating time was recorded automatically. It is the time it takes the tool from getting the input to producing output(s).

F1-score is calculated using following rules: TP: *actual operation set* $\wedge$ *predicted operation set*; FP: *predicted operation set* - *actual operation set*; FP: *actual operation set* - *Predicted operation set*. In the study on type 1 output, F1-score of each sample could be gotten directly. In the study on type 2 output, because *actual operation set* only had operations on each attribute in one direction, but *predicted operation set* had recommended operations in both directions, so we extended *actual operation set* in the other direction according to *n entity set*. Then we calculated F1-score of each output and kept the one with maximum value as F1-score of the sample.

5.2 Results and Analyses

5.2.1 RQ1: Supportable range

Among 59 common attributes, 86.4% (51/59) of them could be supported by the tool. The main reason why an attribute couldn't be supported was the values of the attribute couldn't be read or modified by VSTO.

We answer research question RQ1 as follows:

The tool can support most of PPT operations.

5.2.2 RQ2: Effectiveness

Type 1 output. In the study on type 1 output, we obtained 58 samples after removing 2 invalid samples. The overall average F1-score was 88.01%.

As shown in Figure 5 (a) and Figure 5 (b), there is no obvious monotonic relationship between F1-score and the number of operations in the input or the number of entities on the page.

Figure 5 (c) reveals that tag attributes perform better than numeric attributes. The average F1-score of the former is 90.73%, while that of the latter is only 81.02%.

It is encouraging that among the 58 samples, the recall of 56 samples is 100%, which means our recommended operations can cover almost any subsequent operation. And the average precision is only 82.01%, which shows that the main problem in type 1 output of our tool is to provide some extra recommended operations, or it can be said that out tool provide some extra output entities. A possible reason is that we remove the attributes that the input entity doesn't exist in attribute selection, which may lose some useful information. For example, an entity labeled e_i is the input entity without text and e_j is an entity with text. But the relevance doesn't include this difference because the attributes related to text have been removed in the previous phase. Therefore, the relevance between e_j and e_i is artificially high and e_j may become an extra output entity.

Type 2 output. In the study on type 2 output, we obtained 21 samples after removing 3 invalid samples. The overall average F1-score was 72.14%.

Figure 6 shows that F1-score decreases with the increase of the number of the attributes whose values are uniform within entity set (different from the values of the entities outside the set) and it's mainly due to the decrease in precision, which means some extra recommended operations are given. At this time, the reason for extra output operations is mainly because our tool provides more attributes to be modified than the user actually needs. For example, there are three attributes whose values are uniform within a highly relevant entity set and different from the values of the input entity. It's very common that the user wants the input entity and the highly relevant entity set aligned with each other on the

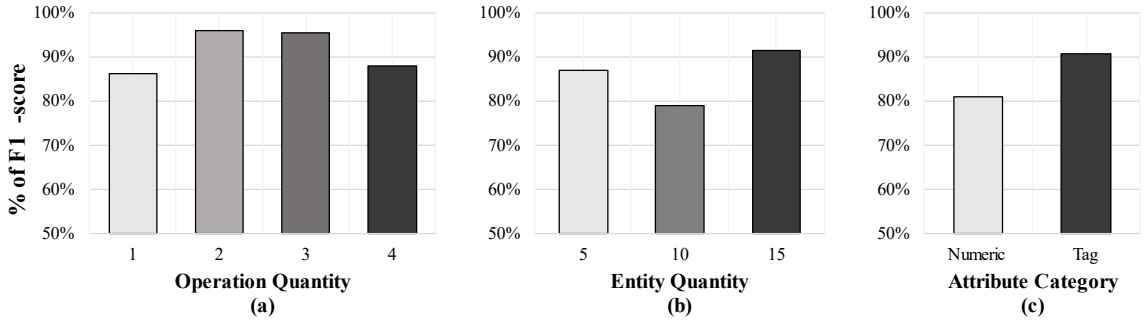


Figure 5: Type 1 output's average F1-score for different operation quantities, entity quantities, attribute categories.

first two attributes but remains the difference on the third attribute. However, out tool will provide recommended operations in all three attributes and that's probably why F1-score decreases with the increase of the number of the attributes.

We answer research question RQ2 as follows:

The tool provides suggestions with a F1-score of 72.14-88.01% and they cover most of subsequent transformations. These results indicate that TIDY is effective enough to support smart transformations for entity consistency in PPT files.

5.2.3 RQ3: Overhead

Type 1 output. In the study on type 1 output, the time to produce suggestions was 6.8ms on average. Figure 7 (a) shows that 84.5% (49/58) of the samples took less than 5ms.

Type 2 output. In the study on type 2 output, the time to produce suggestions was 8.6ms on average. As shown in Figure 7 (b), 81.0% (17/21) of the samples took less than 5ms.

We answer research question RQ3 as follows:

The tool took only marginal time overhead of several milliseconds to generate suggestions, which shows that TIDY is efficient enough to support smart transformations for entity consistency in PPT files.

5.3 Threats to Validity

The 15 attributes we used in the studies on RQ2 and RQ3 may not be representative of all attributes. We mitigate this threat by selecting common and widely applicable attributes by reading popular introductory book of PowerPoint.

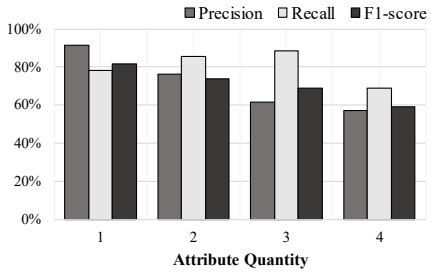


Figure 6: Type 2 output's average Precision, Recall, F1-score for different attribute quantities.

The sample pages may be designed too simple and subjectively affected. We mitigate this threat by using PowerPoint templates. Nowadays, PowerPoint templates are widely used for they contain almost all the page layouts that people may need. PowerPoint templates have many applicable occasions and styles. Some templates are gorgeous and complicated (at the same time without losing order and neatness) while some templates are quite simple like the ones that come with PowerPoint Our sample pages are derived from popular downloads in a large template website (https://www.tukuppt.com/ppt/). After downloading these templates, we may add or delete some entities, then determine the modifiable attribute set according to the specific page.

There is no need to worry about the tool being affected by the participants for its parameters are fixed. But the participants might be affected by the researchers and the tool. Therefore, we only

provide participants with text-based requirements and the researchers won't be on the spot during the study, which avoids researchers' possible unconscious hints. In addition, participants will complete the operations all by themselves instead of being provided with recommended operations which may change their original ideas.

Even if all the requirements are met (monitored by the tool), participants still have great freedom in the choosing entities (and attributes in the third study), which may lead to some unreasonable operations, even if they are not intentional. So, before the study, participants were told that their operations would be judged and they could add some text descriptions for others to understand, which would make them more cautious. After completing all of their own operations, ever participants would be asked to judge the other two participants' operations (through screenshots and

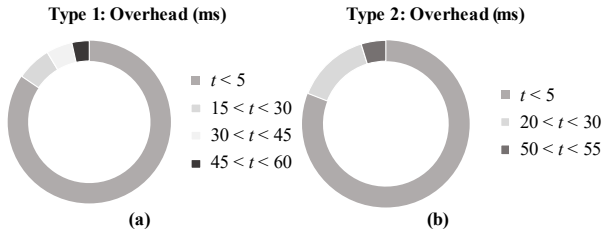


Figure 7: The time overhead of getting Type 1 output (a) and Type 2 output (b).

possible text descriptions). Only the pages whose operations were judged to be reasonable by both two other participants were valid. In order to avoid rejecting operations of other participants just because they were different from the operation of himself on the same sample, the three participants wouldn't share any sample. And whenever participants rejected some operations of a page, they needed to declare a negative reason, which made them work harder to understand others' ideas. In addition, they were informed that there were some pages full of unreasonable operations generated by the researcher, which could avoid them from blindly judging pages as reasonable.

6 RELATED WORK

As a sub-field of program synthesis, PBE enables people to obtain an intended program by simply providing some examples, unlike traditional program synthesis, which needs specifications described by logical formulas [8, 9, 10]. PBE has been applied in many domains and has produced mature industrial products such like Flash Fill in Microsoft Office [4].

This paper presents a PBE-based framework to support smart transformations for entity consistency and it can be applied in PowerPoint. It's related to the entity transformation task, a typical class of PBE task. Most existing works dealing with the task synthesize a single transformer and apply it to all entities [3, 4, 11] while some can handle entity relations [12, 13, 14, 15]. However, as far as we know, no existing work can accomplish the same mission as we do for richly formatted documents like PPT files. We discuss the existing work related to ours in the following.

Compared with its widespread use in spreadsheets, the application of PBE in PowerPoint is not much. The work proposed by Raza et al. address repetitive formatting in PowerPoint [16], which share similar motivation with our single-step framework. Most of their tasks have strict selection criteria on entities, which need more examples to identify specific tasks. Therefore, their approach is more applicable to the entire document. Our work focuses on relevance between entities, rather than specific criteria. And we may provide a few sets of recommendations and identify specific task by user's feedback. So, our approach needs fewer examples, which makes it more suitable for a PPT page.

Blue-Pencil proposed by Miltner et al. [7] is also enlightening to us. It is a modeless system for synthesizing edit suggestions which also doesn't require examples provided in a special mode. Blue-Pencil has been instantiated successfully in several domains such like SQL code and spreadsheets but it cannot handle richly formatted documents such like PowerPoint. What's more, it can identify related examples within noisy action traces. That is also the direction of our future research for we can only deal with noise-free operation history at present.

7 CONCLUSION AND FUTURE WORK

In this paper, we propose a framework with two levels (single-step, multi-step), for recommending subsequent entities and operations based on previous operations and context. The main idea of single-step framework is to get those entities which are relevant with the input entity as candidate output entities and then obtain the output according to the result of the judgement of the output type. Multi-step framework identifies user's intentions based on noise-free operation history, and then performs a similar calculation process as single-step framework. In addition, we developed a PowerPoint add-in based on single-step framework and get satisfactory evaluation results.

Our future work will focus on how to identify user's intentions based on operation history with noise. In addition, we expect to instantiate and evaluate our framework for more application domains.

REFERENCES

[1] Sumit Gulwani. 2016. Programming by Examples (and its Applications in Data Wrangling). In *Verification and Synthesis of Correct and Secure Systems*. IOS Press.

[2] Sumit Gulwani, Oleksandr Polozov, Rishabh Singh. 2017. Program synthesis. *Foundations and Trends in Programming Languages*. 4(1-2) (2017) 1–119.

[3] Vu Le and Sumit Gulwani. 2014. FlashExtract: A Framework for Data Extraction by Examples. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, 542–553.

[4] Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. In *Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. 317–330.

[5] Rishabh Singh and Sumit Gulwani. 2012. Learning semantic string transformations from examples. *Proc. VLDB Endow*. 5(8), 740-751.

[6] Reudismam Rolim, Gustavo Soares, Loris D'Antoni, Oleksandr Polozov, Sumit Gulwani, Rohit Gheyi, Ryo Suzuki, Björn Hartmann. 2017. Learning syntactic program transformations from examples. In *Proceedings of the 39th International Conference on Software Engineering*. 404–415.

[7] Anders Miltner, Sumit Gulwani, Vu Le, Alan Leung, Arjun Radhakrishna, Gustavo Soares, Ashish Tiwari, Abhishek Udupa. 2019. On the Fly Synthesis of Edit Suggestions. In *Proceedings of the ACM on Programming Languages*. Volume 3: 143:1-143:29.

[8] Saurabh Srivastava, Sumit Gulwani, Jeffrey S. Foster. 2010. From program verification to program synthesis. In *Proceedings of the 37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. 313-326.

[9] Shachar Itzhaky, Sumit Gulwani, Neil Immerman, Mooly Sagiv. 2010. A simple inductive synthesis methodology and its applications. In *Proceedings of the 25th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*. 36–46.

[10] Sumit Gulwani, Susmit Jha, Ashish Tiwari, Ramarathnam Venkatesan. 2011. Synthesis of loop-free programs. In *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation*. 62–73.

[11] Navid Yaghmazadeh and Christian Klinger. 2016. Synthesizing Transformations on Hierarchically Structured Data. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 508–521.

[12] Jiarong Wu, Yanyan Jiang, Chang Xu, ShingChi Cheng, Xiaoxing Ma, Jian Lu. 2018. Poster: Synthesizing Relation-Aware Entity Transformation by Examples. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion*. 266-267.

[13] Sai Zhang and Yuyin Sun. 2013. Automatically synthesizing SQL queries from input-output examples. In *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*. 224–234.

[14] Chenglong Wang, Alvin Cheung, Ras Bodik. 2017. Synthesizing Highly Expressive SQL Queries from Input-Output Examples. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 452–466.

[15] Azza Abouzied, Dana Angluin, Christos Papadimitriou, Joseph M. Hellerstein, Avi Silberschatz. 2013. Learning and verifying quantified Boolean queries by example. Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. 49-60

[16] Mohammad Raza, Sumit Gulwani, Natasa Milic-Fraylin. 2014. Programming by Example using Least General Generalizations. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*. 283-290.