A Logical Investigation on Global Reading of Diagrams (Technical Note)

Ryo Takemura[*] Atsushi Shimojima[†] Yasuhiro Katagiri[‡]

February 21, 2012

Abstract

In graphical or diagrammatic representations, not only a basic component of a diagram, but also a collection of multiple components can form a semantic unit, and it often helps reasoning with that diagram. For example, a row and a column in a table, or transitively closed nodes in a directed graph can be regarded as a semantic unit on its own. Designers have long noticed the importance of information conveyed by these global objects [1, 2, 21, 3], and considerable psychological research has also been done on their interpretation process [12, 22, 4, 13]. In this paper, we investigate the global reading of diagrams from the viewpoint of logic. We call those abstract global objects that are invariant under any representation in the given diagrammatic system as "units," and we define units in a table, a directed graph, and an Euler diagram, respectively. We give a mathematical characterization of them, and study relationships between them. We further investigate how units in a diagram help logical reasoning with that diagram.

1 Introduction

By "global objects," we mean those patterns or structures in diagrams that allow the extraction of higher-level information about the represented domain. Examples are a "cloud" consisting of multiple dots in a scatter plot that allows an estimation of the correlation strength of two variables [8], an "ascending staircase" made of multiple columns in a vertical bar graph that allows the observation of an increasing trend [12], rows of cells in a feature-list table that allow comparison of the features of different printer models [18], sequences of edges in a subway connection map that allow the extraction of the set of stations reachable from a particular station [17], and a group of closed contours in a Euler diagrams that allows the extraction of the set of subsets of a particular group. The extraction of such higher-level information has been variously called "macro reading" [21], "pattern perception" [3], "direct translation" [12], and "cognitive integration" [13], and contrasted to the extraction of more concrete information from local objects, such as individual dots (scatter plot), bars (bar charts), cells (feature-list tables), edges (subway connection maps), and regions (Euler

[*]Department of Philosophy, Keio University, Japan.
takemura@abelard.flet.keio.ac.jp
[†]Faculty of Culture and Information Science, Doshisha University, Japan.
ashimoji@mail.doshisha.ac.jp
[‡]Department of Complex and Intelligent Systems, Future University Hakodate, Japan.
katagiri@fun.ac.jp

diagrams). Although both designers and researchers agree that the former largely accounts for the inferential advantages of information graphics [1, 2, 21, 12, 22, 4, 3, 13], few attempts have been made to flesh out what exact computational advantages it provides.

We call the extraction of higher-level information from diagrams "global reading," and investigate it from the viewpoint of logic. One of the difficulties in studying global reading is to define which collection of components of a diagram is regarded in general as a meaningful unit, i.e., a global object. Given a diagram, we may find a variety of global objects that could be interpreted and used to help reasoning. For example, if a table of entries is given, a collection of objects in a single row or in column, objects on the diagonal, or even four objects in the center of the table could have specific meaning in the table. In general, any collection of objects in a table can be regarded as a global object depending on the specific use or the particular presentation of the table. As a first step toward understanding this rich potential of global reading, we confine ourselves to an investigation of one of the most abstract global objects—not all those that are visually meaningful, but those that are invariant under any representation in the given diagrammatic system. We call these invariant global objects "units," and define units for systems of tables, directed graphs, and Euler diagrams, as they are broadly conceived. Then, we give a mathematical characterization of units of our diagrams, and study relationships between these units.

The idea is to compare, from a mathematical point of view, different diagrammatic systems for (1) their potentials of global reading and (2) the computational advantages resulting from them. For (1), we define a common set of basic data and, for representations of the common data, we introduce abstract syntax for respective diagrammatic systems. The common basic data and abstract syntax based on them reveal abstract structures of units in different systems, and make it possible to study relationship between them.

As for (2), we apply the well-developed complexity analysis in computer science. Although theories of computability or tractability in logic and computer science have been applied to reveal general limitations of human cognitive capacities and human computational tractability [9, 14, 6], few studies have dealt with how forms of data representation and concrete algorithms affect the complexity of computation. Investigations on algorithms and data structures (such as arrays, lists, and trees etc.), however, are well-developed in the literature on computer science. (See, for example, [16].) Our work is unique in applying this aspect of computer science to the analysis on actual human reasoning that exploit units in diagrams. Our general finding is that units in a diagram considerably reduce the number of steps required in searching objects by putting some objects together, which helps reasoning tasks associated with that diagram.

The definition of the basic data set is given in Section 2. This section also describes the general strategy of our computational analysis. Sections 3–5 are then devoted to the systems of tables, directed graphs, and Euler diagrams, respectively. In Section 3, we define an abstract syntax for our tables, in which an entire row or column is defined as a unit in a table. We show that these units considerably reduce searching steps required in reasoning tasks associated with tables. In Section 4, we define an abstract syntax for our directed graphs, which are called P-graphs in this paper. A unit in a P-graph is defined as the set of nodes reachable from a node x, denoted by $\mathsf{reachfrom}(x)$, which is shown, in view of the theory of ordered sets, to correspond to the principal filter generated by x. We show that units in our directed graphs help in such reasoning tasks that tables does not work effectively. In particular, our units in P-graphs perform well in such reasoning tasks that transversing directed edges upwardly. In Section 5, we introduce an abstract syntax for our Euler diagrams,

and define a unit as the set of circles and points that are inside a circle x. It is shown that the unit, denoted by $\mathsf{circle}(x)$, corresponds, in view of the theory of ordered sets, to the principal ideal generated by x, which is the dual notion of the principal filter. We also show that, since our units in Euler diagrams are the dual of units in P-graphs, they work effectively in tasks that have dual forms with respect to the successful tasks with P-graphs. That is, our units in Euler diagrams perform well in such reasoning tasks that extracting inclusion or subset relations between circles downwardly. For use in the reviewing process, we list in Appendix A all of our inference tasks and their analyses, some of which are omitted in the text.

2 Basic data

We start by defining basic data from which our diagrams (tables, directed graphs, and Euler diagrams) are constructed. We also specify typical reasoning tasks associated with the basic data.

2.1 Basic data set

Our basic data are $a\!:\!x$ (meaning "a is x") and $a\not:\!x$ ("a is not x").

Definition 2.1 A **basic data set** is $S = (A \times X, R^{+,-})$, where:

- A and X are disjoint sets, for which the respective elements are called "objects" and "properties."
- R^+ and R^- are disjoint subsets of $A \times X$, i.e., $R^+ \cap R^- = \emptyset$.

  Each element $(a, x) \in R^+$ (resp. $(a, x) \in R^-$) is sometimes called a *positive (resp. negative) data* for which we write $a\!:\!x$ (resp. $a\not:\!x$).

Notationally, we use $\Box$ to denote a data without specifying some information thereof. For example, $a\!:\!\Box$ denotes some data $a\!:\!x$, $a\!:\!y$, and/or $a\!:\!z$, etc., and $a\Box x$ denotes either $a\!:\!x$ or $a\not:\!x$.

Note that we do not assume $R^+ \cup R^- = A \times X$ for our data set. In particular, if this condition holds, such a basic data set is said to be **complete**, otherwise **partial**.

Remark 2.2 If we exclude the condition of disjointness between R^+ and R^-, inconsistent data are allowed to our basic data sets.

A basic data set S is usually identified with the set $R^+ \cup R^-$, and it is represented by a sequence of data as in Example 2.3.

Example 2.3 (Basic data set) Let $A = \{a, b, c, d\}, X = \{x, y, z, w\}$, and let:

$R^+ \cup R^- = \{a\!:\!x\ ,\ c\!:\!z\ ,\ b\!:\!z\ ,\ a\not:\!w\ ,\ a\!:\!z\ ,\ d\!:\!y\ ,\ c\not:\!x\ ,\ d\!:\!w\ ,\ a\not:\!y\ ,\ c\not:\!w\ ,\ d\not:\!x\ ,\ b\not:\!w\ ,\ d\!:\!z\ ,\ c\!:\!y\}$

2.2 Reasoning tasks involving basic data

To investigate how units in a diagram help in performing reasoning tasks, we define typical reasoning tasks about our basic data.

Compared with usual sentential/symbolic reasoning, in diagrammatic reasoning, not only constructions of and operations on diagrams, but also comprehension or recognition of (resulting) diagrams plays an essential role. Hence, as an important basic step toward the computational analysis on diagrammatic reasoning, we concentrate in this paper on the analysis on comprehension of diagrams, and leave a more thorough analysis for future work.

Our reasoning tasks are enumerations of some data from a given sequence of basic data or a given diagram. We classify our tasks depending on whether: (1) it is in a positive form or a negative form; (2) it states a relationship between objects and properties, or between properties and properties; (3) it requires to enumerating subjects or predicates:

Positive forms:

Between objects and properties:

(a is $\vec{x}$) Enumeration of all properties that an object a has.

(All of $\vec{a}$ are x) Enumeration of all objects that satisfy a property x.

Between properties:

(x is $\vec{x}$) Enumeration of all properties that are implied by x.

(All of $\vec{x}$ are x) Enumeration of all properties each of which implies x.

Negative forms:

Between objects and properties:

(a is not $\vec{x}$) Enumeration of all properties that an object a does not have.

(None of $\vec{a}$ are x) Enumeration of all objects each of which does not have x.

Between properties:

(No x is $\vec{x}$) or equivalently (None of $\vec{x}$ are x) Enumeration of all properties each of which is incompatible with x.

In computational complexity analysis studied in computer science, a Turing machine is commonly used as a model of computation, and algorithms are usually described in terms of a Turing machine or a specific programming language. However, we are interested in how our diagrams help actual human reasoning, and hence, we consider an imaginary person as our computational model, and we design our algorithms to closely reflect as much as possible actual reasoning.

Let us consider an imaginary person who is asked to solve our reasoning tasks. We assume that he has a poor memory, that is, he can only retain a single datum at any one time, i.e., cannot retain more than one datum. For example, he cannot retain three data, say $a : x, b : y, c : z$, at once, and hence, he cannot look for such data in a single search. He also cannot compare two collections of data at once. However, we assume that he can enumerate a number of data that are reduced to a single key (i.e., condition) such as $\square : x$ at once in a single search. We further assume that he is given only a sequence of data or a diagrammatic representation of these, and he is only allowed to write a final answer but not allowed to take any notes. Thus, he cannot delete or modify any part of the given diagram.

Under these conditions, it turns out that his possible operations in solving our tasks are reduced to the following two operations: (1) focusing on a unit by searching the given diagram; and (2) enumerate/output some target data by searching the given sequence of data or the focused unit in the diagram. These operations mainly consist of "searching," and hence, we measure complexity of our algorithm for a given task in terms of the number of data to be searched, or equivalently the number of steps required, in searching for target data.

In this paper, we are interested in the *worst case*, i.e., the amount of time an algorithm would take on the worst possible input configuration.

As our algorithms are given based only on combinations of the above searching operations, our complexity analyses are completely converted into those in terms of a Turing machine or a specific programming language.

Example 2.4 (Enumeration of objects) Let us enumerate all objects that satisfy a property y in the data set given in Example 2.3. For this task, we simply look through the sequence of data from the left to right, and enumerate all data of the form $\square : y$, i.e., $c : y$ and $d : y$.

In what follows, since the worst-case of a given partial data set is equivalent to that of a complete data set, we assume, without loss of generality, a complete data set $S = (A \times X, R^{+,-})$ is given, for which cardinalities are $card(A) = l$ and $card(X) = k$ (hence $card(A \times X) = l \times k$).

The most basic and intuitive algorithm to enumerate target data from a given sequence of data is the so-called sequential search or linear search. Given a basic data set formed from a sequence of $l \times k$ data, we look through the sequence from the left to right. Hence, the **search space**, i.e., the number of data to be searched in a single search, or equivalently the number of steps required in a single search, of a basic data set is $l \times k$ in general.

One of the remarkable features of our basic data set is the difficulty to read off relationships between properties.

Algorithms to solve our tasks with a given sequence of data and computational complexities of them are as follows.

(a is $\vec{x}$) We enumerate all data of the form $a : \square$ by looking through the given sequence of $l \times k$ data. Hence, we need the following number of steps at worst:

$$l \times k.$$

Similarly for **All of $\vec{a}$ are x**.

(x is $\vec{x}$) We check $(x \Rightarrow x_1) \wedge \cdots \wedge (x \Rightarrow x_k)$, i.e., $\forall x_i \forall a (a : x \Rightarrow a : x_i)$.

We repeat the following steps for every x_i with $1 \leq i \leq k$.

We repeat the following steps for every a_j with $1 \leq j \leq l$.

1. We look for $a_j : x$ in the given sequence of $l \times k$ data. If we cannot find it, by skipping the next step, we go to case a_{j+1}, unless $j = l$, in which case we go to case x_{i+1} by outputting x_i.
2. We look for $a_j : x_i$ in the given sequence of $l \times k$ data. If we cannot find it, we go to case x_{i+1}. If we find it, we go to case a_{j+1}, unless $j = l$, in which case we go to case x_{i+1} by outputting x_i.

$$(((l \times k) \times 2) \times l) \times k$$

Similarly for **All of** $\vec{x}$ **are** x, and **No** x **is** $\vec{x}$.

(a **is not** $\vec{x}$**)** We enumerate all data of the form $a\not/\square$ by looking through the given sequence of $l \times k$ data. (Hence, essentially the same as a **is** $\vec{x}$.)

$$l \times k$$

Similarly for **None of** $\vec{a}$ **are** x.

3 Tables

While our basic data set is discrete and has no unit, we consider, in a table, each collection of data in a row or column as a unit.

3.1 Construction of tables

Our table is defined as an $A \times X$-matrix over $R^+ \cup R^-$ of a basic data set, that is, a rectangular arrangement of basic data in which rows and columns are indexed by sets A and X, respectively. As usual, any pair of tables, say T_1 and T_2 over $R^+ \cup R^-$, are identified if all values (i.e., data) of ax-entries of T_1 and T_2 are identified, that is, for every fixed $a \in A$ (resp. $x \in X$), values of x-entries, i.e., the row a (resp. a-entries, i.e., the column x) are identified. Hence, each collection of data in a row or in a column is invariant under any particular representation of a table. In contrast, other collections of data in a representation of a table such as a collection of data in a diagonal, or that of four data in the center of the table can vary with every representation of the table. Hence, we regard each collection of data in a row and in a column as a unit in a table.

Definition 3.1 A **table** $T = (A \times X, R^{+,-})$ for a basic data set $S = (A \times X, R^{+,-})$ is an $A \times X$-matrix in which an ax-entry (a, x) is in R^+ (resp. in R^-, or blank) if so is in S.
A **unit of a table** is the following set $\mathsf{col}(x)$ or $\mathsf{row}(a)$ for every $x \in X, a \in A$:

$$\mathsf{col}(x) = \{(a, x) \in R^+ \cup R^- \mid a \in A\}, \qquad \mathsf{row}(a) = \{(a, x) \in R^+ \cup R^- \mid x \in X\}.$$

Example 3.2 (Table) A table for the basic data set of Example 2.3 is represented as in Fig. 1, where two units $\mathsf{col}(y)$ and $\mathsf{row}(a)$ are emphasized.

		x	y	z	w
			$\mathsf{col}(y)$		
$\mathsf{row}(a)$	a	$a:x$	$a\not/y$	$a:z$	$a\not/w$
	b			$b:z$	$b\not/w$
	c	$c\not/x$	$c:y$	$c:z$	$c\not/w$
	d	$d\not/x$	$d:y$	$d:z$	$d:w$

Fig. 1 Partial table

	x	y	z	w
a	$a:x$	$a\not/y$	$a:z$	$a\not/w$
b	$b:x$	$b:y$	$b:z$	$b\not/w$
c	$c\not/x$	$c:y$	$c:z$	$c\not/w$
d	$d\not/x$	$d:y$	$d:z$	$d:w$

Fig. 2 Complete table

Note that some entries of our table may be blank. A table that has some blank entries is called a **partial table**, otherwise a table is called a **complete table**. A complete table is illustrated in Fig. 2.

3.2 Reasoning tasks involving tables

Let us consider the following example of a reasoning task involving a table.

Example 3.3 (Enumeration of objects) Let us enumerate all objects that satisfy a property y in the table given in Fig. 1 of Example 3.2. We first look for and focus on the column y among 4 columns x, y, z, w. We then enumerate all data of the form $\square : y$, i.e., $c : y$ and $d : y$ by looking through all 3 data contained in the column y.

Since the worst-case of a given partial table is equivalent to that of a complete table, we assume without loss of generality that a complete table $T = (A \times X, R^{+,-})$ is given in what follows.

In general, for reasoning tasks involving a table, we first focus on a row (resp. a column) by looking through all l rows (resp. k columns), and then enumerate target data from all k data of the row (resp. l data of the column). Note that, by the nature of matrices, rows and columns are clearly distinguished, that is, to pick out a row from a given table, we need to look through only rows, and not through columns. Hence, the search space of a table is generally limited to $l + k$ compared with $l \times k$ of a basic data set.

As with a basic data set, it is not easy to read off relationships between properties by using a table.

Algorithms for our tasks with tables and their complexities are as follows.

(a is $\vec{x}$) (1) By looking through l rows $a_1, \ldots, a_l$, we focus on the row a. (2) Then, we enumerate all data of the form $a : \square$ by looking through k data of the unit $\mathsf{row}(a)$. Hence, the number of steps we need at worst is:

$$l + k$$

Similarly for **All of $\vec{a}$ are x**, **a is not $\vec{x}$**, and **None of $\vec{a}$ are x**.

(x is $\vec{x}$) By focusing on the unit $\mathsf{col}(x)$, we compare each data of $\mathsf{col}(x)$ with that of $\mathsf{col}(x_i)$ by moving laterally, i.e., by focusing on $\mathsf{row}(a_j)$.

1. By looking through k columns, we focus on the column x.
2. We repeat the following steps for every x_i with $1 \leq i \leq k$.

   We repeat the following steps for every a_j with $1 \leq j \leq l$.

   (a) We look for the data $a_j : x$ in the unit $\mathsf{col}(x)$ containing l data. If we cannot find it, by skipping the next step, we go to case a_{j+1} unless $j = l$, in that case, we go to case x_{i+1} by outputting x_i.

   (b) We look for the data $a_j : x_i$ in the unit $\mathsf{row}(a_j)$ containing k data. (Note that there is no need to look for the unit $\mathsf{row}(a_j)$ as we have already focused on it at the time we find $a_j : x$ in the previous step.) If we cannot find it, we go to case x_{i+1}. Otherwise, we go to case a_{j+1} unless $j = l$, in that case we go to case x_{i+1} by outputting x_i.

$$k + (((l + k) \times l) \times k)$$

Similarly for **All of $\vec{x}$ are x**, and **No x is $\vec{x}$**.

4 Directed graphs

As seen in the previous section, it is not easy to read off relationships between properties from tables. This is because the matrix representation of a table is unsuitable to express such relationships. In contrast, we are able to express these explicitly by using directed edges in directed graphs.

4.1 Construction of P-graphs

Our directed graphs are essentially so-called Hasse diagrams for partially-ordered sets. A Hasse diagram is usually defined as the transitive reduct of the directed graph obtained by regarding the ordering relation $\leq$ in a partially-ordered set as the directed edge $\rightarrow$. In this paper, however, we simply keep all transitive edges without taking the transitive reduct, and we call such a directed graph a "P-graph," since it is convenient for our purpose to make our directed graph be perfectly identical with the underlying partially-ordered set.

To represent a data set by a P-graph without any loss of given information, we assume the given data set to be complete. If we try to represent a partial data set by interpreting each data $a\!:\!x$ as $a \rightarrow x$, both a negative data of the form $a \not: x$ and a blank data are equally expressed by the absence of $\rightarrow$-edges. That is, a P-graph is not sufficient to represent a partial data set exhaustively. Although this difficulty can be avoided by introducing another type of edge to express negative data, we here simply restrict a given data set to complete one. Hence, if $a \rightarrow x$ is missing from an instance of a P-graph, then $a \not: x$.

Among the basic properties of directed graphs or partially-ordered sets, the transitivity of $\rightarrow$-edges has played a central role in view of diagrammatic reasoning studies. Transitive inferences exploiting $\rightarrow$-edges in a directed graph can be considered to require no troublesome inference steps, or even immediate steps. Thus, we define a unit in a P-graph as a transitively-closed set of nodes, i.e., for each node x, the set $\mathsf{reachfrom}(x)$ of nodes that are reachable from x by traversing $\rightarrow$-edges. It is shown that, in view of the theory of ordered sets, the set $\mathsf{reachfrom}(x)$ is exactly the principal filter (or upset) generated by x. Furthermore, $\mathsf{reachfrom}(a)$ for $a \in A$ is isomorphic to the set of positive data contained in $\mathsf{row}(a)$ of the corresponding table.

For P-graphs, there may be various other candidates for units. For example, a path in a P-graph can be considered as a unit. However, a path is not closed under transitivity in the usual sense as we will see in Proposition 4.3. Furthermore, for each node x in a P-graph, there can be a large number of paths from x in general, which considerably increases the search space of target data, and can cause computational difficulty. Hence, we do not adopt paths as units here. Among various candidates of units in a P-graph, the dual of our unit $\mathsf{reachfrom}(x)$, i.e., the set $\mathsf{reachto}(x)$ of nodes that are reachable to x, can be worthy of consideration. This set is exactly the principal ideal (or downset) generated by x. Although we agree the importance and the mathematical naturalness of the set, we reserve it for a unit in an Euler diagram in Section 5. We note that, although the diversity of candidates for units in a P-graph shows the versatility of directed graphs, it can cause computational difficulty in reasoning, as we are required to select an appropriate unit among various possibilities.

Definition 4.1 A **P-graph** for a complete data set $(A \times X, R^{+,-})$ is $P = (A \cup X, \rightarrow)$, where $A \cup X$ is the set of nodes, and the $\rightarrow$-edge is defined as follows:

$$a \rightarrow x \quad \textit{iff} \quad (a, x) \in R^{+}$$

$x \to y$ *iff* for all $a \in A$, if $(a, x) \in R^+$ then $(a, y) \in R^+$.

A **unit in a P-graph** P is the set $\mathsf{reachfrom}(s)$ of nodes which are *reachable from* $s \in A \cup X$, i.e.,

$$\mathsf{reachfrom}(s) = \{x \in X \mid s \to x \text{ in } P\}.$$

We also consider the dual of $\mathsf{reachfrom}$, although we do not regard it as our unit, i.e., the set $\mathsf{reachto}(x)$ of nodes which are *reachable to* x:

$$\mathsf{reachto}(x) = \{s \in A \cup X \mid s \to x \text{ in } P\}.$$

A P-graph is exactly an ordered set when we regard the $\to$-edge as an ordering relation.

Lemma 4.2 *$\to$ is a reflexive and transitive ordering relation over X.*

Although it is not essential in this paper, we can postulate $a \to a$ for all $a \in A$.

Proposition 4.3 (Transitive closure and filter) *For every $s \in A \cup X$, $\mathsf{reachfrom}(s)$ is closed under transitivity. That is, $x \in \mathsf{reachfrom}(s)$ and $x \to y$ in P imply $y \in \mathsf{reachfrom}(s)$. Hence, $\mathsf{reachfrom}(s)$ is the principal filter generated by s, and dually, $\mathsf{reachto}(s)$ is the principal ideal generated by s.*

The following proposition shows a correspondence between P-graphs and tables:

Proposition 4.4 (P-graphs and tables) *Let P and T be a P-graph and a table, respectively, for a common complete data set. There are one-to-one correspondences:*

1. *between $\mathsf{reachfrom}(a)$ in P and $\mathsf{row}^+(a) = \{(a, x) \in R^+ \mid x \in X\}$ in T for every $a \in A$: $\mathsf{reachfrom}(a) \simeq \mathsf{row}^+(a)$,*
2. *between the set $\mathsf{reachto}^I(x)$ of "initial nodes" that are reachable to x in P and $\mathsf{col}^+(x) = \{(a, x) \in R^+ \mid a \in A\}$ in T for every $x \in X$: $\mathsf{reachto}^I(x) \simeq \mathsf{col}^+(x)$.*

Proof. (1) is immediate by the following equations. (2) is similar.

$$\mathsf{reachfrom}(a) \;=\; \{x \in X \mid a \to x \text{ in } P\} \;\simeq\; \{(a, x) \in R^+ \mid x \in X\} \;=\; \mathsf{row}^+(a)$$

■

Example 4.5 (P-graph) A P-graph corresponding to the table of Fig. 2 in Example 3.2 is represented as the following Fig. 3, in which $\mathsf{reachfrom}(a)$ (and $\mathsf{reachto}(y)$) is made explicit by a (resp. dotted) circle.

4.2 Reasoning tasks involving P-graphs

Let us consider the following typical reasoning task, in which a P-graph is useful.

Example 4.6 (Enumeration of properties) Let us enumerate all properties that an object a has in the P-graph given in Fig. 3 of Example 4.5. We focus on the node a by looking through all 8 nodes. We then enumerate all nodes reachable from a, i.e., x and z.

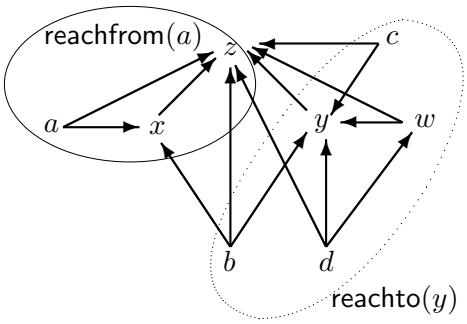


Fig. 3 P-graph

In general, for reasoning tasks involving a P-graph, we first look through all $l+k$ nodes, or equivalently units, and focus on the target. Then, we enumerate target data by searching the unit consisting at most of k nodes. Hence, the typical search space of a P-graph is $l+k+k$. Note that it is presumed that, unlike tables, there is no clear distinction between initial nodes $a, b, c, \ldots$ and non-initial nodes $x, y, z, \ldots$, and hence, the number of units in a P-graph is $l+k$.

In contrast to data sets and tables, it is relatively easy in P-graphs to derive relationships among properties with the help of $\rightarrow$-edges. In particular, since our unit reachfrom is upwardly closed, P-graphs are effective in enumerating predicates of the forms a **is** $\vec{x}$ and x **is** $\vec{x}$, and not as effective in enumerating subjects of the form **All of** $\vec{a}$ **are** x and **All of** $\vec{x}$ **are** x.

Whereas algorithms on data sets and tables work uniformly for both positive and negative forms of our tasks, there are differences in algorithms in P-graphs for these, as only positive relationships are explicitly expressed by $\rightarrow$-edges in P-graphs. Thus, in general, we are required to perform more steps in negative forms of tasks than for positive ones.

Algorithms for our tasks with P-graphs and their complexities are as follows.

(x is $\vec{x}$) (1) By looking through $l+k$ nodes, we focus on the node x. (2) Then, we enumerate a maximum of k nodes of the unit reachfrom(x).

$$l+k+k$$

Similarly for a **is** $\vec{x}$.

(All of $\vec{x}$ are x) We repeat the following steps for each x_i with $1 \leq i \leq k$.

1. By looking through $l+k$ nodes, we focus on the node x_i.
2. By looking through a maximum of k nodes of the unit reachfrom(x_i), we look for x. If we find it, we output x_i.

$$(l+k+k) \times k$$

Similarly for **All of** $\vec{a}$ **are** x, and **None of** $\vec{a}$ **are** x.

(a is not $\vec{x}$) We enumerate all nodes that are not reachable from a.

(1) By looking through $l+k$ nodes, we focus on the node a. (2) For every x_i with $1 \leq i \leq k$, we repeatedly look for x_i in the unit reachfrom(a) containing up to k nodes, and if we cannot find it, we output x_i.

$$(l+k)+(k \times k)$$

If we regard reachto rather than reachfrom as units, the complexity analysis on such P-graphs is equivalent to that for Euler diagrams given in the next section.

5 Euler diagrams

As with P-graphs, we are able to express relationships among properties explicitly by using topological relations between circles in Euler diagrams.

5.1 Construction of Euler diagrams

We introduce an abstract syntax for Euler diagrams in the relation-based framework of Mineshima-Okada-Takemura [10]. In the framework, an abstract Euler diagram is specified by the set of topological (inclusion and exclusion) relations holding between circles and points in the diagram. (See, e.g., [5] for the standard region-based approach, in which a diagram is specified by the set of regions.)

We consider the most basic Euler diagrams consisting only of circles and points, and we do not adopt linking between points (nor linking between diagrams). Thus, our Euler diagrams cannot express disjunctive information about points, and hence, as with P-graphs, we assume a given data set to be complete.

A unit of an Euler diagram is defined as the set of circles and points that are inside a circle x, which corresponds, in view of a P-graph, to the set $\mathsf{reachto}(x)$ of nodes reachable to x. Such a set is adopted as the canonical interpretation of a circle in the syntactic model, which is constructed, following the usual Lindenbaum construction in the literature of algebraic semantics for symbolic logics, to prove the completeness theorem of the Euler diagrammatic inference system in [10]. We further show that the set of points contained in $\mathsf{circle}(x)$ is isomorphic to a set of positive data contained in the column x of the corresponding table.

Definition 5.1 An (abstract) **Euler diagram** for a complete data set $S = (A \times X, R^{+,-})$ is $E = (A \cup X, \mathsf{Rel})$, where $A \cup X$ is the union of sets of abstract circles X and of points A. Rel is the set of **relations** between abstract circles and points, i.e., the *inclusion relation* $\sqsubset$, the *exclusion relation* $\vdash$, and the *crossing (or partial-overlapping) relation* $\bowtie$, such that:

$a \sqsubset x \in \mathsf{Rel}$ *iff* $(a, x) \in R^+$ in S.

$x \sqsubset y \in \mathsf{Rel}$ *iff* for all $a \in A$, if $(a, x) \in R^+$ then $(a, y) \in R^+$.

$a \vdash x \in \mathsf{Rel}$ *iff* $(a, x) \in R^-$.

$x \vdash y \in \mathsf{Rel}$ *iff* there is no $a \in A$ such that $(a, x) \in R^+$and $(a, y) \in R^+$.

$x \bowtie y \in \mathsf{Rel}$ *iff* $x \sqsubset y \notin \mathsf{Rel}$ and $x \vdash y \notin \mathsf{Rel}$.

A **unit in an Euler diagram** E is the following set for every $t \in A \cup X$:

$$\mathsf{circle}(t) = \{s \in A \cup X \mid s \sqsubset t \in \mathsf{Rel}\}$$

In particular, $\mathsf{circle}(a)$ is empty for every point $a \in A$. (Alternatively, we may postulate $a \sqsubset a \in \mathsf{Rel}$ for every $a \in A$, in that case, $\mathsf{circle}(a)$ is the singleton $\{a\}$.)

The following proposition shows correspondences among units of our diagrams.

Proposition 5.2 (Euler diagrams, P-graphs, and tables) *Let E and P be an Euler diagram and a P-graph, respectively, for a common complete data set S. There is a one-to-one correspondence between* $\mathsf{circle}(t)$ *of E and* $\mathsf{reachto}(t)$ *of P for every $t \in A \cup X$:*

$$\mathsf{circle}(t) \simeq \mathsf{reachto}(t).$$

Hence, $\mathsf{circle}(t)$ *is, in view of the theory of ordered sets, the principal ideal generated by* t*. Let* $\mathsf{circle}^P(x)$ *be the set of points inside a circle* x*. Let* T *be the table for* S*. Then, there is a one-to-one correspondence between* $\mathsf{circle}^P(x)$ *of* E *and* $\mathsf{col}^+(x)$ *of* T *for every* $x \in X$*:*

$$\mathsf{circle}^P(x) \ \simeq \ \mathsf{col}^+(x).$$

Our Euler diagrams are represented by concrete plane diagrams as usual.

Example 5.3 (Euler diagram) An Euler diagram corresponding to the P-graph in Fig. 3 of Example 4.5 is represented as in Fig. 4. To indicate the correspondence between units reachto in the P-graph and circle in the Euler diagram, the transitive reduct of the P-graph of Fig. 3 is expressed by dotted arrows.

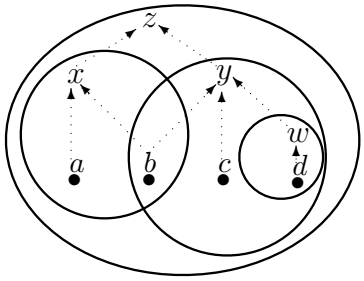


Fig. 4 Euler diagram

5.2 Reasoning tasks involving Euler diagrams

Let us consider the following reasoning task, in which an Euler diagram is useful.

Example 5.4 (Enumeration of properties) Let us enumerate all properties each of which implies y in the Euler diagram given in Fig. 4 of Example 5.3. We focus on the circle y by looking through all 4 circles. We then enumerate all circles that are inside the contour of y, i.e., w.

In general, in a reasoning task involving an Euler diagram, we first focus on a target circle by looking through all k circles, i.e., units. Then, by looking through a maximum of k circles or l points in the unit, i.e., inside the circle, we enumerate target data. Hence, the typical search space of an Euler diagram is $k + k$ or $k + l$. In contrast to the case with P-graphs, we here assume that circles and points are clearly distinguished, and hence, there is no need to look through circles and points altogether. However, if we consider there is no clear distinction among circles and points, then the search space is $(k + l) + (k + l)$.

As with P-graphs, Euler diagrams work effectively in deriving relationships between properties. In contrast to P-graphs, Euler diagrams work effectively in enumerations of subjects of the forms $\vec{a}$ **are** x and $\vec{x}$ **are** x, and not so in enumerations of predicates.

Algorithms for our tasks with Euler diagrams and their complexities are as follows.

(x is $\vec{x}$) We repeat the following steps for each x_i with $1 \leq i \leq k$.

1. By looking through k circles $x_1, \ldots, x_k$, we focus on the circle x_i.
2. By looking through a maximum of k circles of the unit $\mathsf{circle}(x_i)$, we look for the circle x. If we find it, we output the circle x_i.

$$(k + k) \times k$$

If we assume circles and points are not distinguished, we need the following number of steps at worst: $(l + k + l + k) \times k$.

Similarly for a **is** $\vec{x}$, and a **is not** $\vec{x}$.

(All of $\vec{x}$ are x) (1) By looking through k circles $x_1, \ldots, x_k$, we focus on the circle x. (2) We enumerate a maximum of k circles of the unit $\mathsf{circle}(x)$.

$$k + k$$

In the case where circles and points are not distinguished: $(l + k) + (l + k)$.

Similarly for **All of $\vec{a}$ are** x.

(None of $\vec{a}$ are x) (1) By looking through a maximum of k circles, we focus on the circle x. (2) For every a_j with $1 \leq j \leq l$, we look for the point a_j by looking through the unit $\mathsf{circle}(x)$ containing up to l points, and when we cannot find it, we out put a_j.

$$k + (l \times l)$$

In the case where circles and points are not distinguished: $(l + k) + ((l + k) \times l) = (l + k) \times (l + 1)$.

Remark 5.5 We have not discussed Venn diagrams. A Venn diagram is defined as a set of shaded minimal regions (cf. [5]), and hence a unit in a Venn diagram may be a certain set of minimal regions. However, from a cognitive science viewpoint, it is not clear how people use Venn diagrams in actual reasoning tasks (see, e.g., [15] for a discussion), and also from a mathematical viewpoint, it is arguable which collection of minimal regions forms a semantic unit in a Venn diagram (for example, [19] proposes "conjunctive regions" as meaningful collections of regions). Thus, we leave an analysis on Venn diagrams as future work.

6 Conclusion and future work

We investigated global objects in diagrams from the viewpoint of logic, and defined mathematically meaningful global objects as units. The relationship among these units in respective diagrams as well as their well-established mathematical counterparts are summarized as follows:

Tables		P-graphs		Euler diagrams	Mathematical counter part
$\mathsf{row}^+(a)$	$\simeq$	$\mathsf{reachfrom}(a)$			principal filter
		$\updownarrow$ dual			$\updownarrow$ dual
		$\mathsf{reachto}(x)$	$\simeq$	$\mathsf{circle}(x)$	principal ideal
$\mathsf{col}^+(x)$	$\simeq$	$\mathsf{reachto}^I(x)$	$\simeq$	$\mathsf{circle}^P(x)$	(Lindenbaum construction)

We further investigated computational advantages resulting from global reading by applying the complexity analysis in computer science. In general, units in a diagram reduce the number of steps required in a searching of data by putting some data together. Our results are summarized as follows.

Positive forms	Objects-Properties		Properties-Properties	
	Enum. properties a is $\vec{x}$	Enum. objects All of $\vec{a}$ are x	Enum. predicates x is $\vec{x}$	Enum. subjects All of $\vec{x}$ are x
Set	$l \times k$	$l \times k$	$(((l \times k) \times 2) \times l) \times k$	$(((l \times k) \times 2) \times l) \times k$
Table	$\boxed{l + k}$	$\boxed{l + k}$	$k + (((l + k) \times l) \times k)$	$k + (((l + k) \times l) \times k)$
P-graph	$l + k + k$	$(l + k + k) \times l$	$\boxed{l + k + k}$	$(l + k + k) \times k$
Euler	$(k + l) \times k$	$\boxed{k + l}$	$(k + k) \times k$	$\boxed{k + k}$

Negative forms	Objects-Properties		Properties-Properties
	Enum. properties a is not $\vec{x}$	Enum. objects None of $\vec{a}$ are x	Enum. predicates No x is $\vec{x}$, or None of $\vec{x}$ are x
Set	$l \times k$	$l \times k$	$(((l \times k) \times 2) \times l) \times k$
Table	$\boxed{l + k}$	$\boxed{l + k}$	$k + (((l + k) \times l) \times k)$
P-graph	$(l + k) + (k \times k)$	$(l + k + k) \times l$	$((l + k + k + k) \times l) \times k$
Euler	$(k + l) \times k$	$k + (l \times l)$	$k + ((k + ((l + l) \times l)) \times k)$

In our analysis, our units in P-graphs and Euler diagrams work not so effectively for negative forms of tasks, although these diagrams seem to have certain advantages in our actual reasoning. Our result may suggest the possibility that there is another type of units or even another mechanism proper to negative forms of problems, and we leave investigations thereof as our future work.

Our computational analysis was based on the assumption that human visual processing is sequential in identifying objects by their visual features in a visual scene. Further assumptions were derived from it to estimate the number of steps required, namely, (1) that the time it takes to identify an object with a set of visual features is proportional to the number of all the visual objects in the scene, and (2) secondly, the time it takes to enumerate objects which share a set of visual features is proportional to the number of target objects.

Experiments on visual search have revealed that human visual object identification mostly takes time proportional to the number of objects in a visual scene, but that when search targets can be discriminated from distractor objects by one simple feature, e.g., color, shape, line direction etc., they 'pop-out' and can be identified in a fixed amount of time independent of the number of distractors ([11]). Treisman [20] proposed feature integration theory in which a lower level feature-by-feature visual processing and a higher level processing of feature integration by attention traversal were assumed. Klein [7] proposed a mechanism of inhibitory tagging to explain how the visual system works sequentially in visual search by not visiting the same objects multiple times.

As all the visual objects in consideration in this paper, rows and columns in a table, and transitive sets of nodes in a graph as well as points and circles with labels, are complex objects only identifiable by a set of multiple visual features, we believe our sequential processing assumption is mostly in accord with findings in visual processing research. In some cases, however, pop-out phenomena can be effectively utilized when single feature discrimination is possible, e.g., tables with black-white or color-coded entries.

We also assumed that the time it takes to check a set of visual features of an object for identification is constant irrespective of the type and complexity of the features. It could well be much harder to check a connected set of nodes in a complicated web of a graph than to check a column in a table. Identification of such complex objects as a set of nodes in a graph probably depends not only on conjunctive combination of their visual features but

also on the relevance to the demands imposed by whatever problem the human is solving at the time. Research on visual processing have mostly focused on perceptual mechanisms underlying visual object identification and recognition, and interaction between higher level cognitive processes and lower level perceptual processes have not studied extensively. Logical reasoning with diagrams provides us with a good domain to investigate the interaction by integrating logical, computational and psychological approaches.

References

[1] J. Bertin, *Semiology of Graphics: Diagrams, Networks, Maps*, The University of Wisconsin Press, Madison, WI, 1973.

[2] Bertin, Jacques, *Graphics and Graphic Information*, Originally published in France in 1977, Walter de Gruyter, Berlin, 1981.

[3] W. S. Cleveland, *The Elements of Graphing Data*, Hobart Press, Summit, NJ, 1994,

[4] Guthrie, John T. and Weber, Shelley and Kimmerly, Nancy, Searching Documents: Cognitive Processes and Deficits in Understanding Graphs, Tables, and Illustrations, *Contemporary Educational Psychology*, 18, 186–221, 1993.

[5] J. Howse, G. Stapleton, and J. Taylor, Spider Diagrams, *LMS Journal of Computation and Mathematics*, Volume 8, 145-194, London Mathematical Society, 2005.

[6] A. Isaac, J. Szymanik, Logic and Complexity in Cognitive Science, submitted, 2011.

[7] R. M. Klein, Inhibitory tagging system facilitates visual search, *Nature*, 334, 430-431, 1988.

[8] S. M. Kosslyn, *Elements of Graph Design*, W. H. Freeman and Company, 1994.

[9] Hector J. Levesque, Logic and the complexity of reasoning, *Journal of Philosophical Logic*, Volume 17, Number 4, 355-389, 1988.

[10] K. Mineshima, M. Okada, and R. Takemura, A Diagrammatic Inference System with Euler Circles, accepted for *Journal of Logic, Language and Information.*
A preliminary version is available at: http://abelard.flet.keio.ac.jp/person/takemura/index.html

[11] K. Nakayama and G. H. Silverman, Serial and parallel processing of visual feature conjunctions, *Nature*, 320, 264-265, 1986.

[12] Steven Pinker, A Theory of Graph Comprehension, *Artificial Intelligence and the Future of Testing*, Roy Freedle ed., L. Erlbaum Associates, 73–126, 1990.

[13] Ratwani, Raj M. and Trafton, J. Gregory and Boehm-Davis, Deborah A., Thinking Graphically: Connecting Vision and Cognition During Graph Comprehension, *Journal of Experimental Psychology: Applied*, 14, 1, 36–49, 2008.

[14] Iris Van Rooij, The Tractable Cognition Thesis, *Cognitive Science*, Volume 32, Issue 6, 939-984, 2008.

[15] Y. Sato, K. Mineshima, and R. Takemura, The efficacy of Euler and Venn diagrams in deductive reasoning: empirical findings, *Diagrams 2010*, 6-22, 2010.

[16] Robert Sedgewick, *Algorithms*, 2nd Edition. Addison-Wesley, 1988.

[17] Atsushi Shimojima, Derivative Meaning in Graphical Representations, *Proceedings of 1999 IEEE Symposium on Visual Languages*, 212-219, 1999.

[18] Atsushi Shimojima, The Inferential-Expressive Trade-Off: A Case Study of Tabular Representations, *Diagrams 2002*, 116–130, 2002.

[19] Ryo Takemura, Proof-Theoretical Investigation of Venn Diagrams: a Logic Translation and Free Rides, submitted to Diagrams 2012.

[20] A. Treisman, Features and objects in visual processing, *Scientific American*, 254, 11, 114-125, 1986.

[21] E. R. Tufte, *Envisioning Information*, Graphics Press, Cheshire, CT, 1990.

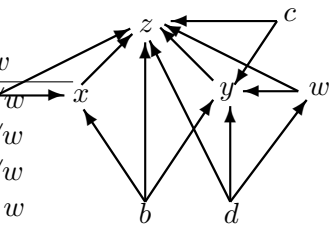
[22] H. Wainer, Understanding Graphs and Tables, *Educational Researcher*, 21, 14-23, 1992.

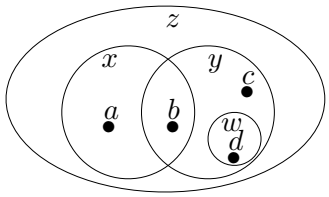
A Computational analysis

In this appendix, we list all of our inference tasks and their analyses.

$R^+ \cup R^- = \{a\!:\!x\ ,\ c\!:\!z\ ,\ b\!:\!z\ ,\ a\not\!:\!w\ ,\ a\!:\!z\ ,\ d\!:\!y\ ,\ c\not\!:\!x\ ,\ d\!:\!w\ ,\ a\not\!:\!y\ ,\ c\not\!:\!w\ ,\ d\not\!:\!x\ ,\ b\not\!:\!w\ ,\ d\!:\!z\ ,\ c\!:\!y\}$

	x	y	z	w
a	$a\!:\!x$	$a\not\!:\!y$	$a\!:\!z$	$a\not\!:\!w$
b	$b\!:\!x$	$b\!:\!y$	$b\!:\!z$	$b\not\!:\!w$
c	$c\not\!:\!x$	$c\!:\!y$	$c\!:\!z$	$c\not\!:\!w$
d	$d\not\!:\!x$	$d\!:\!y$	$d\!:\!z$	$d\!:\!w$





	Objects-Properties		Properties-Properties	
Positive forms	Enum. properties a is $\vec{x}$	Enum. objects All of $\vec{a}$ are x	Enum. predicates x is $\vec{x}$	Enum. subjects All of $\vec{x}$ are x
Set	$l \times k$	$l \times k$	$(((l \times k) \times 2) \times l) \times k$	$(((l \times k) \times 2) \times l) \times k$
Table	$\boxed{l + k}$	$\boxed{l + k}$	$k + (((l + k) \times l) \times k)$	$k + (((l + k) \times l) \times k)$
P-graph	$l + k + k$	$(l + k + k) \times l$	$\boxed{l + k + k}$	$(l + k + k) \times k$
Euler	$(k + l) \times k$	$\boxed{k + l}$	$(k + k) \times k$	$\boxed{k + k}$

	Objects-Properties		Properties-Properties
Negative forms	Enum. properties a is not $\vec{x}$	Enum. objects None of $\vec{a}$ are x	Enum. predicates No x is $\vec{x}$, or None of $\vec{x}$ are x
Set	$l \times k$	$l \times k$	$(((l \times k) \times 2) \times l) \times k$
Table	$\boxed{l + k}$	$\boxed{l + k}$	$k + (((l + k) \times l) \times k)$
P-graph	$(l + k) + (k \times k)$	$(l + k + k) \times l$	$((l + k + k + k) \times l) \times k$
Euler	$(k + l) \times k$	$k + (l \times l)$	$k + ((k + ((l + l) \times l)) \times k)$

A.1 a is $\vec{x}$ (Enumeration of all properties that an object a has)

Data set: We enumerate all data of the form $a\!:\!\square$ by looking through the given sequence of $l \times k$ data. Hence, we need the following number of steps at worst:

$$l \times k$$

Table: (1) By looking through l rows $a_1, \ldots, a_l$, we focus on the row a. (2) Then, we enumerate all data of the form $a : \square$ by looking through k data of the unit $\mathsf{row}(a)$. Hence, we need the following number of steps at worst:

$$l + k$$

P-graph: (1) By looking through $l + k$ nodes $a_1, \ldots, a_l, x_1, \ldots, x_k$, we focus on the node a. (2) Then, we enumerate a maximum of k nodes of the unit $\mathsf{reachfrom}(a)$. Hence, we need the following number of steps at worst:

$$l + k + k$$

Euler: We repeat the following steps for each x_i with $1 \leq i \leq k$.

1. By looking through k circles $x_1, \ldots, x_k$, we focus on the circle x_i.
2. By looking through a maximum of l points of the unit $\mathsf{circle}(x_i)$, we look for the point a. If we find it, we output x_i.

Hence, we need the following number of steps at worst:

$$(k + l) \times k$$

If we assume circles and points are not distinguished, we need the following number of steps at worst:

$$(l + k + l + k) \times k$$

A.2 All of $\vec{a}$ are x (Enumeration of all objects that satisfy a property x)

Data set: (The same as the case of a is $\vec{x}$.)

We enumerate all data of the form $\square{:}x$ by looking thorough the given sequence of $l \times k$ data.

$$l \times k$$

Table: (Essentially the same as the case of a is $\vec{x}$.)

(1) By looking through k columns $x_1, \ldots, x_k$, we focus on the column x. (2) Then, we enumerate all data of the form $\square{:}x$ by looking through l data of the unit $\mathsf{col}(x)$.

$$k + l$$

P-graph: We repeat the following steps for each a_j with $1 \leq j \leq l$.

1. By looking through $l + k$ nodes $a_1, \ldots, a_l, x_1, \ldots, x_k$, we focus on the node a_j.
2. By looking through a maximum of k nodes of the unit $\mathsf{reachfrom}(a_j)$, we look for x. If we find it, we output a_j.

$$(l + k + k) \times l$$

Euler: (1) By looking through k circles $x_1, \ldots, x_k$, we focus on the circle x. (2) We enumerate a maximum of l points of the unit $\mathsf{circle}(x)$.

$$k + l$$

If we assume circles and points are not distinguished, we need the following number of steps at worst:

$$l + k + l + k$$

A.3 x is $\vec{x}$ (Enumeration of all properties that are implied by x)

Data set: We check $(x \Rightarrow x_1) \wedge \cdots \wedge (x \Rightarrow x_k)$, i.e., $\forall x_i \forall a(a{:}x \Rightarrow a{:}x_i)$.

We repeat the following steps for every x_i with $1 \leq i \leq k$.

We repeat the following steps for every a_j with $1 \leq j \leq l$.

1. We look for $a_j : x$ in the given sequence of $l \times k$ data. If we cannot find it, by skipping the next step, we go to case a_{j+1} unless $j = l$, in which case we go to case x_{i+1} by outputting x_i.
2. We look for $a_j{:}x_i$ in the given sequence of $l \times k$ data. If we cannot find it, we go to case x_{i+1}. If we find it, we go to case a_{j+1} unless $j = l$, in which case we go to case x_{i+1} by outputting x_i.

$$(((l \times k) \times 2) \times l) \times k$$

Table: By focusing on the unit $\mathsf{col}(x)$, we compare each data of $\mathsf{col}(x)$ with that of $\mathsf{col}(x_i)$ by move laterally, i.e., by focusing on $\mathsf{row}(a_j)$.

1. By looking through k columns, we focus on the column x.
2. We repeat the following steps for every x_i with $1 \leq i \leq k$.

   We repeat the following steps for every a_j with $1 \leq j \leq l$.

   (a) We look for the data $a_j : x$ in the unit $\mathsf{col}(x)$ containing l data. If we cannot find it, by skipping the next step, we go to case a_{j+1} unless $j = l$, in that case, we go to case x_{i+1} by outputting x_i.

   (b) We look for the data $a_j{:}x_i$ in the unit $\mathsf{row}(a_j)$ containing k data. (Note that there is no need to look for the unit $\mathsf{row}(a_j)$ since we have already focused on it at the time we find $a_j{:}x$ in the previous step.) If we cannot find it, we go to case x_{i+1}. Otherwise, we go to case a_{j+1} unless $j = l$, in that case, we go to case x_{i+1} by outputting x_i.

$$k + (((l + k) \times l) \times k)$$

P-graph: (The same as the case of a is $\vec{x}$.)

(1) By looking through $l + k$ nodes $a_1, \ldots, a_l, x_1, \ldots, x_k$, we focus on the node x. (2) Then, we enumerate a maximum of k nodes of the unit $\mathsf{reachfrom}(x)$.

$$l + k + k$$

Euler: (Essentially the same as the case of a is $\vec{x}$.)

We repeat the following steps for each x_i with $1 \leq i \leq k$.

1. By looking through k circles $x_1, \ldots, x_k$, we focus on the circle x_i.
2. By looking through a maximum of k circles of the unit $\mathsf{circle}(x_i)$, we look for the circle x. If we find it, we output x_i.

$$(k + k) \times k$$

If we assume circles and points are not distinguished, we need the following number of steps at worst:

$$(l + k + l + k) \times k$$

A.4 All of $\vec{x}$ are x(Enumeration of all properties each of which implies x)

Data set: The same as x is $\vec{x}$, i.e., $(((l \times k) \times 2) \times l) \times k$

Table: The same as x is $\vec{x}$, i.e., $k + (((l + k) \times l) \times k)$

P-graph: (Essentially the same as the case of All of $\vec{a}$ are x)

We repeat the following steps for each x_i with $1 \leq i \leq k$.

1. By looking through $l + k$ nodes, we focus on the node x_i.
2. By looking through a maximum of k nodes of the unit $\mathsf{reachfrom}(x_i)$, we look for x. If we find it, we output x_i.

$$(l + k + k) \times k$$

Euler: (Essentially the same as the case of All of $\vec{a}$ are x)

(1) By looking through k circles $x_1, \ldots, x_k$, we focus on the circle x. (2) We enumerate a maximum of k circles of the unit $\mathsf{circle}(x)$.

$$k + k$$

If we assume circles and points are not distinguished, we need the following number of steps at worst:

$$l + k + l + k$$

A.5 a is not $\vec{x}$ (Enumeration of all properties that an object a does not have)

Data set: (The same as the case of a is $\vec{x}$.)

We enumerate all data of the form $a\not\square$ by looking through the given sequence of $l \times k$ data.

$$l \times k$$

Table: (The same as the case of a is $\vec{x}$.)

(1) By looking through l rows $a_1, \ldots, a_l$, we focus on the row a. (2) Then, we enumerate all data of the form $a\not\square$ by looking through k data of the unit $\mathsf{row}(a)$.

$$l + k$$

P-graph: We enumerate all nodes that are not reachable from a.

(1) By looking through $l + k$ nodes, we focus on the node a. (2) For every x_i with $1 \leq i \leq k$, we repeatedly look for x_i in the unit $\mathsf{reachfrom}(a)$ containing up to k nodes, and if we cannot find it, we output x_i.

$$(l + k) + (k \times k)$$

Euler: (The same as the case of a is $\vec{x}$.) We enumerate all circles such that $a \vdash x_i$ holds.

We repeat the following steps for every x_i with $1 \leq i \leq k$.

1. By looking through a maximum of k circles, we focus on the circle x_i.
2. We look for the point a in the unit $\mathsf{circle}(x_i)$ containing up to l points, and if we cannot find it, we output x_i.

$$(k+l) \times k$$

If we assume circles and points are not distinguished, we need the following number of steps at worst:

$$(l+k+l+k) \times k$$

A.6 None of $\vec{a}$ are x (Enumeration of all objects each of which does not have x)

Data set: The same as the case of a is $\vec{x}$, i.e., $l \times k$

Table: The same as the case of a is $\vec{x}$, i.e., $l + k$

P-graph: (The same as the case of All of $\vec{a}$ are x.)

We repeat the following steps for each a_j with $1 \leq j \leq l$.

1. By looking through $l + k$ nodes, we focus on the node a_j.
2. By looking through a maximum of k nodes of the unit $\mathsf{reachfrom}(a_j)$, we look for x, and if we cannot find it, we out put a_j.

$$(l+k+k) \times l$$

Euler: (1) By looking through a maximum of k circles, we focus on the circle x. (2) For every a_j with $1 \leq j \leq l$, we look for the point a_j by looking through the unit $\mathsf{circle}(x)$ containing up to l points, and when we cannot find it, we out put a_j.

$$k + (l \times l)$$

If circles and points are not distinguished: $(l + k) + ((l + k) \times l) = (l + k) \times (l + 1)$

A.7 No x is $\vec{x}$, or equivalently None of $\vec{x}$ are x (Enumeration of all properties each of which x is incompatible with)

Data set: The same as x is $\vec{x}$, i.e., $(((l \times k) \times 2) \times l) \times k$

Table: The same as x is $\vec{x}$, i.e., $k + (((l + k) \times l) \times k)$

P-graph: We check that x and x_i are disjoint.

We repeat the following steps for each x_i with $1 \leq i \leq k$.

We repeat the following steps for each a_j with $1 \leq j \leq l$.

1. By looking through $l + k$ nodes, we focus on the node a_j.
2. By looking through a maximum of k nodes of the unit $\mathsf{reachfrom}(a_j)$, we look for x. If we cannot find it, we got to case a_{j+1} unless $j = l$, in that case, we output x_i and go to case x_{i+1}.

3. By looking through a maximum of k nodes of $\mathsf{reachfrom}(a_j)$, we look for x_i. If we find it, we go to case x_{i+1}. Otherwise, we go to case a_{j+1} unless $j = l$, in that case, we output x_i and go to case x_{i+1}.

$$((l + k + k + k) \times l) \times k$$

Euler: We enumerate all circles that shares no element with x.

1. By looking through a maximum of k circles, we focus on the circle x.
2. We repeat the following steps for each circle x_i with $1 \leq i \leq k$.
   (a) By looking through a maximum of k circles, we focus on x_i.
   (b) We repeat the following steps for each point a_j with $1 \leq j \leq l$.
      i. We look for the point a_j in the unit $\mathsf{circle}(x_i)$. If we cannot find it, we go to case a_{j+1}, unless $j = l$, in that case, we out put x_i and go to case x_{i+1}.
      ii. We look for the point a_j in the unit $\mathsf{circle}(x)$. If we find it, we go to case x_{i+1}. Otherwise, we go to case a_{j+1} unless $j = l$, in that case, we output x_i and go to case x_{i+1}.

$$k + ((k + ((l + l) \times l)) \times k)$$