

## Meta-Pattern Extraction: Mining Cycles \*

Jennifer Seitzer<sup>†</sup>, James P. Buckley, Alvaro Monge

Computer Science Department

University of Dayton

300 College Park

Dayton, Ohio 15469-2160

{seitzer, buckley, monge}@cps.udayton.edu

### Abstract

Inductive computing, comprised of machine learning, data mining, and knowledge discovery, seeks to extract causal patterns from data. Meta-patterns are patterns of extracted patterns. The meta-pattern we study here is the cycle.

In this paper, we illustrate the ubiquity of cycles and argue that as responsible knowledge engineers, we need to identify and actively seek cycles out of data. We present a methodology for cycle mining and exemplify such a pursuit in a generic relational model as well as briefly discussing the authors' implementation, computer system **INDED**. System **INDED** is a machine learning system implementing inductive logic programming techniques of pattern extraction.

### Introduction and Motivation

Knowledge discovery in databases has been defined as the non-trivial process of identifying valid, novel, potentially useful, and understandable patterns in data [5]. A pattern is often denoted in the form of an IF-THEN rule (IF *antecedent* THEN *consequent*), where the antecedent and consequent are logical conjunctions of predicates (first order logic) or propositions (propositional logic) [10]. In [9], the author observes that knowledge can take on more complex forms than a simple implication as a causal chain or network by interconnecting the consequent of one rule to the antecedent of another. *Acyclic graphs* are used extensively as knowledge representation constructs in knowledge discovery in databases

as seen in [6] and [10]. It is largely our goal to bring to the reader's attention the benefit of employing a graph capable of possessing cycles to represent learned knowledge and identifying the appropriate propositional vertices as cycle participants.

### Why Cycles Typically Exist in Data

Much of our social construction of reality stems directly from the physical cycles of the earth revolving around the sun, and its rotation on its axis: the temporal calendar by which we live. We invent and self-impose other cycles based on these: the seasons and holidays of the year, the organization of the academic and fiscal years. These cycles are the basis for much of our common sense, domain knowledge, and they spawn related cycles such as the cycle of seasons justifying the cycle of weather events we might experience.

Human behavior is based on reinforcement, repetition, and routine [7]. This human behavior manifests in the telephone calls and purchases we make, the food we eat, and the hardships, such as illness, we experience. In our speech, we often allude to a negative sequence of human behaviors, such as those manifested as overall poor quality in manufacturing, or more profoundly, violence in our society, as "being caught in a vicious cycle." Or we embrace a positive sequence of behaviors and try to secure a cycle by reinforcing excellent employee performance or by instilling good habits in our children.

Cycles appear in our creative pursuits, as well. An analysis of the tonality of virtually any musical work of Bach will indicate a traversal around the circle of fifths [3]. A study of Escher's art will have the observer entranced in subtle, circular, repetitious patterns [8].

The authors contend that evidence of this cyclic nature of humans and our world exists in our data.

\*Copyright © 1999, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

<sup>†</sup>Partially supported under Grant 9806184 by the National Science Foundation.

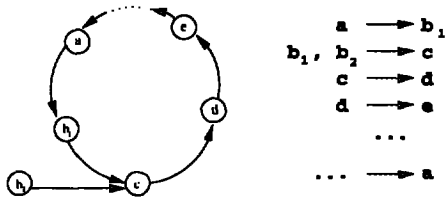


Figure 1: Example of a cycle.

To identify and extract these special patterns can help us better understand and control our environment. Moreover, because events comprising a cycle are interdependent, we are given a choice as to the particular event on which to focus to perpetuate a cycle, and are assured a sense of continuity until the cycle is broken.

### What is a Cycle?

The graph theoretic definition tells us that a cycle is a path in a directed graph that begins and ends at the same node. We represent each logical proposition in the knowledge base as a node.

**Definition 1.1 (cycle)** Let  $G = (V, E)$  be a directed graph where  $u, u', v_0, v_1, \dots, v_k \in V$ . A path of length  $k$  from  $u$  to  $u'$  is a sequence  $(v_0, v_1, \dots, v_k)$  of vertices such that  $u = v_0$  and  $u' = v_k$  and  $(v_{i-1}, v_i) \in E$  for  $i = 1, \dots, k$ . The length of the path is the number of edges in the path. A cycle is a path  $(v_0, v_1, \dots, v_k)$  where  $v_0 = v_k$ ; it is a simple cycle if vertices  $v_1, \dots, v_k$  are distinct [4].

We consider a cycle intuitively as a chain of causal dependencies with feedback. By chain, we mean a collection implications, or rules, where one rule proves the antecedent of another using modus ponens. The feedback manifests by the head (or consequent) of a discovered rule appearing in the body (or antecedent) of a previously discovered rule in which one of its antecedent conjuncts is the head (all other conjuncts are true). Or, through transitive closure, the head appears in the antecedent of a different rule, the head of which participated in the chain of deduction to derive the new antecedent, again, by modus ponens. Figure 1 shows a cycle with its corresponding rules.

In this paper, we use a hypergraph knowledge representation structure discussed in [13] to represent logical implications and dependencies amongst knowledge base propositions. Although the forthcoming cycle detection algorithm works for both simple and general cycles, without loss of generality, we assume all cycles are simple.

For example, consider the cyclic affect advertising has on sales. Assume an inductive learning system for a consumer products company has learned the following patterns or rules (note: the left hand side of  $\rightarrow$  below contains the antecedent, and the right hand side, the consequent):

$brand\_X\_advertises \rightarrow Jones\_visits\_store\_of\_brand\_X$   
 $Jones\_visits\_store\_of\_brand\_X \rightarrow Jones\_buys\_brand\_X$   
 $Jones\_buys\_brand\_X \rightarrow brand\_X\_profits$

The chain of causal dependencies here is simply

$brand\_X\_advertises \downarrow$   
 $Jones\_visits\_store\_of\_brand\_X \downarrow$   
 $Jones\_buys\_brand\_X \downarrow$   
 $brand\_X\_profits$

At this point, assume the learner discovers the rule.

$brand\_X\_profits \rightarrow brand\_X\_advertises$

This newly discovered rule forms a cycle of causal dependencies with a feedback loop connecting the notion of profits and advertising (the more money that is available, the more the company can spend on advertising). At this point, the system deems all four propositions **cycle participants**. Identification of cycle participants provides the user a choice: to break or to perpetuate the cycle. This choice is dependent on the goals of the overall system.

### Why Cycles are Important

The authors in [9] tell us that an isolated IF-THEN pattern or rule is **useful** when it helps to achieve a system goal. Cycles are powerful sets of connected IF-THEN rules for two related reasons. If the system goal is dependent on a cycle participant, then the user can choose to attain the system goal by perpetuating the cycle. To perpetuate, the user can activate the antecedent of any participating IF-THEN rule. One antecedent fires all of the other participating IF-THEN rules by  $k$  applications of modus ponens. This means that the user may *choose* which event or scenario to activate. In essence, because of the cyclic causality, activating one cycle participant is equivalent to activating all  $k$  scenarios. In the above example, the company producing  $brand\_X$  need only activate one cycle participant – such as putting up a billboard – so as to assert the fact  $brand\_X\_advertises$  to activate the cycle of profit acquisition. Similarly, if the goals of the system warrant that the cycle be broken, only one causal link must be broken to break the circle of fire.

The second powerful aspect of a cycle is its inherent implication of continuity. Above, it may be

all well and good that company X acquires a profit from Jones's purchase one time. It is probably safe to say, however, that the underlying goal of most consumer products companies is to continue to sell to customers, and hence continue to bring in a profit. Extraction of a cyclic pattern alerts a system that targeting any of the cycle participant activities, assures continuous attainment of the goal, at least until the cycle is broken.

We have implemented cycle detection work in the machine learning area of inductive logic programming. Any inductive pursuit including data mining and knowledge discovery, however, can be used to acquire the constituent IF-THEN patterns of cycles. We, therefore, present a formal model exemplifying cycle detection in a more traditional relational database setting. We then summarize results attained from the ILP implementation. Regardless of the specific inductive pursuit, however, we contend that a cycle detection algorithm should be invoked regularly as the knowledge base continues to grow, in order to exploit the benefits of cycle detection.

### A Methodology for Discovering Cycles

Discovery of cycles has as much to do with the knowledge representation structure as it does with the rule discovery algorithm itself. The detection algorithm operates by performing a depth first search on a graph. We assume a hypergraph representation where logical predicates appearing as heads of rules are represented as vertices, and sets of (conjoined) predicates, appearing as rule bodies, are represented as hyperedges. For example, a rule  $P(X) \text{ --- } R(X), \dots, Q(Y)$ , with head  $P(X)$  and body  $R(X), \dots, Q(Y)$  is internally represented as vertex  $P(X)$  with incoming hyperedge  $R(X), \dots, Q(Y)$ .

Our methodology consists of four steps:

1. Discover constituent IF-THEN patterns.
2. Insert constituent IF-THEN patterns into the dependency hypergraph knowledge structure described above
3. Perform the cycle identification algorithm. As each rule is learned, the graph is traversed in depth-first manner and if a cycle is introduced, all cycle participants are identified and the cycle is output.

This step exploits a standard algorithm for cycle detection which is presented in [2] and employs postorder numbering of propositions as variables, as shown below.

4. Determine the current state of the knowledge base. As the dependency graph grows with each new pattern, the current state of the combined collection is reevaluated. Each time, a (possibly) new set of **true** facts or propositions is generated.

Note: for any given cycle, either all or none of the cycle participants will be included in the current state (i.e., either all or none of the constituent rules will be fired).

**Algorithm 1.2 (Cycle Detection Algorithm)**  
If the domain knowledge is initially acyclic, our structure embodies a forest of causal trees.

**Input:** Hypergraph representation  $\mathcal{P}$   
of knowledge base

Newly discovered rule of the form  $h \text{ --- } b$

**Output:** Set of cycle participants if cycle exists

*BEGIN ALGORITHM 1.2*

for each discovered rule  $h \text{ --- } b$

insert rule into  $\mathcal{P}$

With head of new rule  $h$  do

perform an iterative dfs(h)

- as each proposition is popped,

assign its postorder number

if postorder number of head is  $\geq$  any of its

incoming edge postorder nums,

then there exists a causal cycle

formed by the incoming edge source

node to the head, and the path from the head

to the source of the incoming edge.

*END ALGORITHM 1.2*

### Formal Model

We exemplify our method using a standard relational model where rules of the form

$$X_1 \text{ --- } Y_1, \dots, Y_m$$

are discovered. Rules possess individual subgoals  $Y_i (1 \leq i \leq m)$  which are conjoined (conjunction denoted by comma) forming the *body* of the rule. The consequent  $X_1$  forms the *head* of the rule.

We define a database  $D$  as a set of relations  $D = \{R_1, \dots, R_l\}$  where each relation  $R_i$  is a set of ordered  $n$ -tuples of the form  $\langle d_1, \dots, d_n \rangle$  where each  $d_j$  is in the domain  $D_j$ , for  $1 \leq j \leq n$ . The cardinality of  $R_i$  is the number of  $n$ -tuples in the relation. We make the following assumptions originally presented in [12]:

- *Closed-world Assumption.* Information not contained in the database is assumed *false*.

- *Unique-Name Assumption.* Any item in the database schema has a unique name; items with different names are different.
- *Domain-Closure Assumption.* There are no other individual items than those in the database.

A database instance  $r$  is the set of tuples in the database at a given moment in time. To discover any rule, a particular instance of a database is used. A *classification rule* is a rule of the form:

$$(X_1 = b_1) \leftarrow (Y_1 = a_1), \dots, (Y_n = a_n)$$

where individual subgoals equate specific values to attributes, forming the conjuncted rule body, as well as the consequent  $X_1$  being formed by such an equation. We say that attributes  $Y_1 \dots Y_n$  determine, or classify, the value of  $X_1$ . These rules are derived from a given instance of the database with the associated support and confidence measures as defined in [11]:

- the *support* for a rule,  $C_2 \leftarrow C_1$  is the percentage of tuples that satisfy  $C_1 \wedge C_2$
- the *confidence* for a rule,  $C_2 \leftarrow C_1$  is the percentage of tuples that satisfy  $C_2$  given all tuples that satisfy  $C_1$ .

## Mining a Rule

Let  $r_1$  be the following instance of database  $D' = \{A, B, C, D, E\}$ .

A	B	C	D	E
$a_1$	$b_1$	$c_1$	$d_5$	$e_1$
$a_2$	$b_1$	$c_2$	$d_5$	$e_2$
$a_3$	$b_1$	$c_3$	$d_5$	$e_3$
$a_4$	$b_2$	$c_4$	$d_3$	$e_4$
$a_5$	$b_1$	$c_5$	$d_5$	$e_5$
$a_6$	$b_1$	$c_6$	$d_3$	$e_6$
$a_7$	$b_1$	$c_7$	$d_5$	$e_7$
$a_8$	$b_1$	$c_8$	$d_5$	$e_8$
$a_9$	$b_1$	$c_9$	$d_5$	$e_9$
$a_{10}$	$b_1$	$c_{10}$	$d_5$	$e_{10}$

Applying a standard classification rule mining algorithm such as that found in [11], we obtain the rule

$$(D = d_5) \leftarrow (B = b_1)$$

with confidence 88% and support 80%. This rule will be used to predict this causal relation over all instances of  $D'$ .

## Mining a Cycle

The process of mining an entire cycle may be an incremental one. That is, the entire cycle may be detected in more than one database instance. As database instances change, new patterns are learned that may be cycle participants. Assume that new instances  $r_2, r_3$ , and  $r_4$  of the above database are used to extract additional causal patterns. In particular, assume the classification rule

$$(A = a_7) \leftarrow (D = d_5)$$

from instance  $r_2$ , and the pattern

$$(B = b_1) \leftarrow (A = a_7)$$

from instance  $r_4$ . Also assume these rules are extracted with confidences 70% and 85% and with supports 95% and 91%, respectively. We now have the meta-pattern of a cycle in database  $D$ . The cycle can be denoted

$$((B = b_1), (D = d_5), (A = a_7), (B = b_1))$$

We use individual metrics of each constituent rule to quantify the strength of the cycle. These metrics also facilitate determining the starting point of the meta-pattern for future predictive use.

**Definition 1.3 (cycle confidence)** *The confidence of a cycle is the minimum confidence value of any of the constituent rules forming the cycle.*

Because detection of cycles provides the user an option to continue or break any given cycle, we quantify the probability of a cycle continuing without any intervention.

**Definition 1.4 (support)** *The support of a cycle is the minimum support value of any of the constituent rules forming the cycle.*

To invoke a cycle, we identify the aspect of the state (a rule antecedent in the form of a logical proposition) which most likely will activate the cycle.

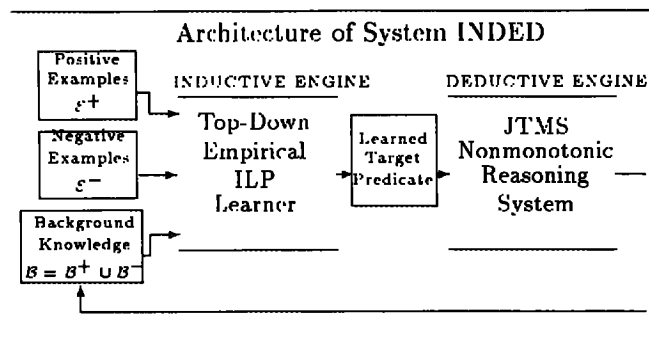
**Definition 1.5 (starting point)** *The starting point of a cycle  $C$  is the antecedent of rule  $r'$  where  $r'$  has maximum support of all constituent rules forming the cycle.*

The above cycle has confidence 70% and support 80% and has starting point  $(D = d_5)$ .

## Implementation in Inductive Logic Programming

The authors have implemented the cycle detection method in an inductive logic programming system **INDED**(pronounced "indeed") which performs

both INDuction to learn rules comprising an intensional knowledge base, and DEDuction to compute the current state – part of the extensional knowledge base [13]. Although the syntax of the generated rules differs slightly, the meta pattern of a cycle remains the same. That is, the cycle is formed by a circular sequence of rule heads firing subsequent rules.



By providing small data sets for the positive and negative examples, intensional rules, and extensional facts of the background knowledge, we were able to learn cycles in the geneology domain. We are currently examining the cyclic relationship of stress (indicated quantitatively by blood pressure) and a mononucleosis diagnosis (indicated by titers being above an acceptable level). We are also studying and experimenting with diagnosis of Lyme Disease because of its puzzling, circular, and seemingly inconsistent symptom set. Early results have been promising and will be included in a future publication.

### Future Work

Along with continued experimentation, we expect to expound on the theory of learned meta-patterns. In particular, we are studying *near cycles*, chains of deduction which could possibly be forced to form a complete cycle if certain conditions were satisfied. We are examining cycles that are made up of propositions with alterability coefficients, cycles involving other types of rules including association rules [1], as well as considering overlapping cycles.

### Conclusion

A cycle mining methodology and cycle detection algorithm have been presented. Both of these involve dependency hypergraph knowledge representation in the form of logical implications.

The authors posit that just as unknown patterns exist in data, unknown cycles of patterns also exist.

By identifying these meta-patterns we gain more insight, and hence, more control, over the environment being examined. By detecting the meta-pattern of a cycle, we may better predict and control the action to be taken to achieve the encompassing system's goal. That is, we may choose to perpetuate or break the cycle.

### References

- [1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Bulletin*, pages 207-216, May 1993.
- [2] Alfred V. Aho and Jeffrey D. Ullman. *Foundations of Computer Science*. Computer Science Press, 1995.
- [3] Manfred F. Bukofzer. *Music in the Baroque Era*. W.W. Norton and Company, 1947.
- [4] Corman, Leiserson, and Rivest. *Introduction to Algorithms*. McGraw-Hill, 1991.
- [5] Usama Fayyad and Evangelos Simoudis. Knowledge discovery in databases. Tutorial at IJCAI-95, 1995.
- [6] Ronen Feldman and Ido Dagan. Knowledge discovery in textual databases (kdt). In *Proceedings of the First International Conference on Knowledge Discovery & Data Mining*, pages 112-117, 1995.
- [7] Daniel Goleman. *Emotional Intelligence*. Bantam Books, 1995.
- [8] Douglas R. Hofstadter. *Goedel, Escher, Bach: An Eternal Golden Braid*. Vintage Books, 1980.
- [9] Piatetsky-Shapiro and Frawley, editors. *Knowledge Discovery in Databases*, chapter Knowledge Discovery in Databases: An Overview. AAAI Press/ The MIT Press, 1991.
- [10] J. R. Quindlan. Induction of decision trees. *Machine Learning*, 1:81-106, 1986.
- [11] Raghu Ramakrishnan. *Database Management Systems*. McGraw-Hill, Inc., 1998.
- [12] R. Reiter. Towards a logical reconstruction of relational database theory. *On Conceptual Modeling*, pages 163-189, 1984.
- [13] Jennifer Seitzer. Stable ILP: Exploring the added expressivity of negation in the background knowledge. In *Proceedings of the Frontiers in Inductive Logic Programming Workshop*. Fifteenth International Joint Conference on Artificial Intelligence, Nagoya, Japan, 1997.