Closure and Causality
(A working paper)

John L. Pfaltz,

Dept. of Computer Science, Univ. of Virginia
Charlottesville, VA 22904-4740
`jlp@virginia.edu`

Abstract. We present a model of causality which is defined by the intersection of two distinct closure systems, $\mathcal{I}$ and $\mathcal{T}$. To present empirical evidence to demonstrate that this model has practical validity we then examine computer trace data to reveal causal dependencies between individual code modules. From over 498,000 events in an open source system, we tease our 66 apparent causal dependencies. Finally, we explore how to mathematically model the transformation of the causal topology resulting from unfolding events.

1 Closure Systems

Let $\mathbf{U}$ denote some universe of elements. A **closure system**, $\mathcal{C}$, is any collection of subsets $X, Y, \ldots Z \subseteq \mathbf{U}$, including $\mathbf{U}$ itself, which is closed under intersection. Subsets in $\mathcal{C}$ are said to be **closed**. If $\mathbf{U} = \{a, b, c, d, e\}$ then the collection of closed sets

$$\mathcal{C}_{\mathbf{1}} = \{\emptyset, \{a\}, \{b\}, \{ab\}, \{bd\}, \{abc\}, \{abd\}, \{abce\}, \{abcde\}, \{abcdef\}\} \tag{1}$$

is a closure system.

A closure system can equivalently be defined as $(\mathbf{U}, \varphi)$, where φ is a **closure operator** satisfying four axioms. For all $Y, Z \subseteq \mathbf{U}$,

C1: $Y \subseteq Y.\varphi$,

C2: $Y \subseteq Z$ implies $Y.\varphi \subset Z.\varphi$, and

C3: $Y.\varphi.\varphi = Y.\varphi$.

By C1, $\mathbf{U}$ itself must be closed. Here we are using a suffix operator notation, as we will throughout this paper. Read $Y.\varphi$ as "Y closure". A set Y is **closed** if $Y = Y.\varphi$. It is not hard to show that these two definitions of closure are equivalent.

A closure operator/system can satisfy other axioms depending on the mathematical discipline. A **topological closure** is closed under union, or

C4: $(Y \cup Z).\varphi = Y.\varphi \cup Z.\varphi$.

The closure operator of linear systems, often called the spanning operator, satisfies the Steinitz-MacLane **exchange axiom**

C5: if $p, q \notin Y.\varphi$ and $q \in (Y \cup \{p\}).\varphi$ then $p \in (Y \cup \{q\}).\varphi$.

Such closure systems are called **matroids**. Still other closure operators may satisfy an **anti-exchange axiom**

C6: if $p, q \notin Y.\varphi$ and $q \in (Y \cup \{p\}).\varphi$ then $p \notin (Y \cup \{q\}).\varphi$.

These closure operators, which include the geometric convex hull operator, are said to define **anti-matroid** closure systems. An important antimatroid property is that:

Proposition 1. *Let* $\mathbf{U}$ *be finite, and let* φ *be antimatroid. If* $X.\varphi = Y.\varphi = Z$ *closed, then* $(X \cap Y).\varphi = Z$.

Proof. If $X \cap Y = X$ or Y, the result is trivial. So there exists $p \in X - Y$ and $q \in Y - X$. Now, suppose $(X \cap Y).\varphi \neq Z$, then $(X \cap Y \cup \{p\}).\varphi = Z = (X \cap Y \cup \{q\}).\varphi$ (if not let $X \cap Y \cup \{p\} = X'$ and $X \cap Y \cup \{q\} = Y'$ and repeat the argument) contradicting C6. $\square$

To see why finiteness is required, consider $\mathbf{U} = \mathbf{Z}^+$, and let X be the even integers, Y be the odd integers, and φ be downset closure. Readily $X.\varphi = Y.\varphi = \mathbf{Z}^+$; but $X \cap Y = \emptyset$.

Let $(\mathbf{U}, \varphi)$ be a closure system. Containment, $\subseteq$, forms a natural partial ordering on the closed subsets $\mathcal{C}_1$ shown in (1), and it is well known that the closed sets in $(\mathbf{U}, \varphi)$ so ordered form a lower semi-modular lattice.[1] Figure 1 illustrates the lattice of closed sets of the closure system $\mathcal{C}_1$ ordered by inclusion.

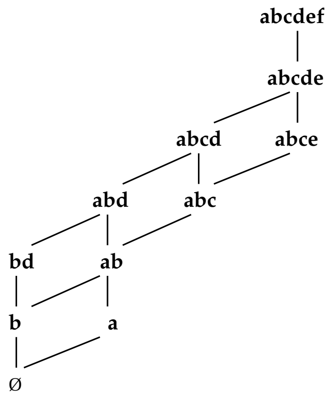

Fig. 1. The lattice of closed sets of $\mathcal{C}_1$

1.1 Generators

Let Z be a closed set. A subset $X \subseteq Z$ is said to generate Z if $X.\varphi = Z$. By C4, every closed set Z generates itself. But, that tends to be uninteresting. We say a generator X of Z is **non-trivial** if $X \subset Z$. In fact, we are really only interested in *minimal* non-trivial generators. For a familiar example, consider a convex polytope which is generated by its extreme vertices under convex hull closure. If φ is antimatroid these minimal elements will be unique (as we show immediately below). More specifically, if X is the minimal generating set, it is called the **generator** of Z, and denoted by $Z.\gamma$. When a closed set may have more than one generator, the collection of all minimal generating sets we denote by $Y.\Gamma = \{Y.\gamma_1, \ldots, Y.\gamma_n\}$ [4]. When there is only a single generating set for any closed set Z we say that $(\mathbf{U}, \varphi)$ is **uniquely generated**.

Proposition 2. *A finite closure system* $(\mathbf{U}, \varphi)$ *is antimatroid if and only if it is uniquely generated.*

[1] This lower semimodularity of closed subsets partially ordered by inclusion has been repeatedly discovered by many authors. See Monjardet [5] for an interesting summary.

Proof. If $(\mathbf{U}, \varphi)$ is not antimatroid then there exists some closed set $Y.\varphi$ with $p, q \notin Y.\varphi$ such that $p \in (Y \cup q).\varphi$ and $q \in (p \cup Y).\varphi$. Then p and q are members of distinct generators of $(Y \cup p).\varphi = (Y \cup q).\varphi$. The converse is similarly shown. $\square$

This proposition can also be treated as a corollary of Prop. 1. A closure operator φ is said to be **finitely generated** if all the generators of Z, $Z.\gamma$ are finite. In this case, Propositions 1 and 2 can be relaxed slightly to read "If $\mathbf{U}$ is finitely generated, and ... ". Nevertheless, from now on we simply assume all closure spaces are finite.

If $(\mathbf{U}, \varphi)$ is antimatroid, then the collection of all sets X_i with the same closure $Z = X_i.\varphi$ constitute a Boolean lattice with $Z.\gamma \subseteq X_i \subseteq Y$. In particular, if $X_i, \ldots X_k$ generate Z then $X_i \cap \ldots \cap X_k$ also generates Z.

2 A Model of Causality

By **causality**, we mean that whenever an event a occurs, event b must subsequently occur. More accurately, whenever a sufficient set A of antecedent events occur then the resultant event b must subsequently occur. This we symbolize by $A \Rightarrow b$. Because of the imperative "must", our sense of causality is deterministic. Readily, A may "cause" a large set of resultant events B, but there is no loss of generality in considering only singleton results.

In this paper we regard an **event** as a primitive element. Thus, we are only considering discrete, or quantum, causality. In a particular system $\mathcal{S}$, it may be that whenever $\mathcal{S}$ is in state s_i and condition c_k is observed then $\mathcal{S}$ makes the transfer t_m from s_i to s_j. In our model the state s_i, condition c_k, and action a_m are all regarded as "events" and the causal relationships could be symbolized by $s_i, c_k \Rightarrow a_m$ and $s_i, t_m \Rightarrow s_j$. Thus, in some respects, this model can be seen as being rather coarse. We indicate a refinement in Section 5.

Let our universe $\mathbf{U}$ be the collection of all possible events $\{e_i\}$. We assume that the resultant event $b \in \mathbf{U}$ must temporally follow each of the antecedent events $a_i \in \mathbf{U}$. Consequently there is an implicit partial order on $\mathbf{U}$ which defines the first "temporal" closure space, $\mathcal{T}$. For a closure operator, we will use the downset $\downarrow$ operator, which we denote by φ_R because we draw temporal dependency left to right in our figures. Thus in Figure 2, event a **temporally precedes** events d, e, h, i or $\{a\}.\varphi_R = \{adehi\}$.[2] Also one can define an upset, $\uparrow$ or φ_L, operator as well as an interval,

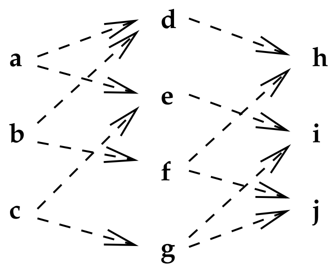

Fig. 2. Ten events partially ordered with respect to time.

or "convex", closure φ_C on any partial order. Convex closure provides a finer grained closed set structure that can be desirable. All three closure operators, φ_L, φ_R and φ_C are antimatroid [9].

We will use the closure space $\mathcal{T} = (\mathbf{U}, \varphi_{\mathcal{R}})$ to model the temporal aspect of causality.

The second closure system $\mathcal{I}$ models the imperative nature of deterministic causality. An implication in a first order logical system has the form

$$(\forall o)[(a(o) \wedge b(o)) \vee c(o) \ \rightarrow \ (d(o) \wedge e(o)] \tag{2}$$

which we read as "for all observations o, if either event a and event b occur or event c occurs then events d and e must also occur". We express (2) more concisely as

$$ab \vee c \ \rightarrow de \tag{3}$$

Here concatenation denotes the logical "and". The precedent P of any implication $P \ \rightarrow \ Q$ may be in disjunctive normal form; but we only allow conjuncts for the consequent Q. Since we are modelling deterministic causality, this leads to no loss of generality.

Consider the following collection of implications on the 10 events of Figure 2.

a b → d	d → h
b → e	e → h
b ∨ c → f	f → i
c → g	f ∨ g → j

They define a closure system $\mathcal{I}$ consisting of the sets $\mathcal{I} = \{abcdefghi, abdefghij,\ acdefghij,$ $bcdefghij, abdefhij, acefghij, acdfghij, bdefghij, cdefghij, adeghij, bdefhij, cefghij, cdfghij, acfgih, deghij,$ $cfgij\}$ and all intersections of these 16 sets. This is admittedly not very intuitive. However, we can graph these 8 implications as in Figure 3. Here the ligature joining arrows (a, d) and (b, d) denotes

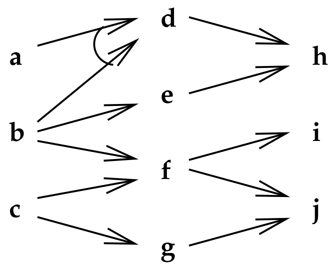

Fig. 3. Logical implications associated with the events of Fig. 2

the logical *and* of the first implication. Implication is transitive, so the figure captures such derived implications as $ab \ \rightarrow \ defhij$ and $c \ \rightarrow \ fgij$. While this graphic representation seems intuitive, it is unable to accurately capture all the nuances present in a collection of implications. We use it only as a suggestive illustration.

[2] It is our habit to denote set-valued operators by this suffix dot notation. Later it will provide a nice lexical distinction with element valued functions represented with the usual prefix notation.

Our thesis is that any set which is closed in both the $\mathcal{T}$ and $\mathcal{I}$ closure systems is a closed set in the causal closure system $\mathcal{C}$. A clearer way of stating this is "if $A \;\; \rightarrow b$ and if for all $a_i \in A$, $b \in \{a_i\}.\varphi_R$ then $A \;\Rightarrow\; b$. That is, the occurrence of all events a_i will cause event b". Superimposing Figure 2 on Figure 3 we get Figure 4. Here we are saying that because the event c logically implies

$$\begin{tikzcd}
 & \mathbf{d} \arrow[dr, Rightarrow] & \\
\mathbf{a} \arrow[ur, Rightarrow] & & \mathbf{h} \\
 & \mathbf{e} & \\
\mathbf{b} \arrow[uuur, Rightarrow] \arrow[dr, Rightarrow] & & \mathbf{i} \\
 & \mathbf{f} \arrow[dr, Rightarrow] & \\
\mathbf{c} \arrow[dr, Rightarrow] & & \mathbf{j} \\
 & \mathbf{g} \arrow[ur, Rightarrow] &
\end{tikzcd}$$

Fig. 4. Causal dependencies reflected by Figures 2 and 3

event g, and g temporally follows c, the occurrence of c must cause the occurrence of g. In this example, both events a and b must occur to cause event d.

Is this reasonable? In the next section we provide empirical evidence that this model has some objective validity.

3 Empirical Evidence

Examples of deterministic causality are hard to find in the real world. This is not necessarily an indication that the real world is not deterministic. Rather, it is usually a manifestation of (a) the presence of errors in observation and/or measurement, or else (2) the presence of complex, conjunctive antecedents whose nature we poorly understand. One exception is computer software.

For their own independent research [16, 17], my colleagues Jinlin Yang and David Evans instrumented the transaction management core of the open source *JBoss* 1.4.2 regression analysis package.[3] They collected 1,227 trace data sequences of observed executions, $\mathcal{O}$, recording 498,489 distinct invocations of 144 different sub-modules $\mathcal{M}$. It is these module invocations that we treat as events.

In a preprocessing step, each module invocation was assigned a unique identifying integer. Figure 5 illustrates a short segment in one trace with the assigned integers. All subsequent steps were then performed on sequences of integers such as ... 3 2 4 1 2 1 2 3 2 4 5 2 ...[4]

The 1,227 observed sequences $\mathcal{O}$ were converted to a $1,227 \times 144$ binary matrix where $(i, k) = 1$ if module $m_k \in \mathcal{M}$ was invoked at least once in the observed trace $o_i \in \mathcal{O}$. (That m_k may have been invoked repeatedly in the trace o_i is ignored in this phase.) This binary matrix denotes a relation R between the trace observations $\mathcal{O}$ and module invocations $\mathcal{M}$, to which we can apply the Galois

[3] *Jboss* is an open source, professional middleware company which can be accessed at `www.jboss.com`.

[4] The usual numeric ordering has no significance here, except that the preprocessor encountered one module invocation before another, possibly in a different unrelated trace. Any other set of 144 distinct symbols would have served our purpose.

```
3    TxManager.getTransaction()Ljavax/transaction/Transaction;
2    TxManager.getThreadInfo()Lorg/jboss/tm/TxManager$ThreadInfo;
4    TxUtils.isActive(Ljavax/transaction/Transaction;)Z
1    TxManager.getStatus()I
2    TxManager.getThreadInfo()Lorg/jboss/tm/TxManager$ThreadInfo;
1    TxManager.getStatus()I
2    TxManager.getThreadInfo()Lorg/jboss/tm/TxManager$ThreadInfo;
3    TxManager.getTransaction()Ljavax/transaction/Transaction;
2    TxManager.getThreadInfo()Lorg/jboss/tm/TxManager$ThreadInfo;
4    TxUtils.isActive(Ljavax/transaction/Transaction;)Z
5    TxManager.suspend()Ljavax/transaction/Transaction;
2    TxManager.getThreadInfo()Lorg/jboss/tm/TxManager$ThreadInfo;
```

Fig. 5. A representative fragment of a trace sequence

closure operator [6]. The Galois closure has been employed by Rudolf Wille in the development of "formal concept analysis" [15], and is best explained in [2].

Briefly, the Galois closure on $\mathcal{O} \times \mathcal{M}$ induces contravariant closure systems on $\mathcal{O}$ and $\mathcal{M}$ in which O_i closed in $\mathcal{O}$ and M_i closed in $\mathcal{M}$, called a concept (O_i, M_i) in [2], indicates that every module invocation $m_k \in M_i$ occurs in every observed trace $o_n \in O_i$. Moreover, M_i is the maximal set of module events occurring in every trace o_n of O_i. Because of this property we can interpret the closure operator on a set of generators as a logical implication. That is, $(\forall o_n \in O_i)[M_i.\gamma \rightarrow M_i]$. This is not a universally quantified implication. It is only quantified over all observations so far, in which the generating events $M_i.\gamma$ are present.

Galois closure can be a useful tool in data mining [3, 14]. At the Univ. of Virginia, we have developed a DDDM (discrete deterministic data mining) software system that has had conspicuous success in several deterministic data mining venues [12, 13]. This system was used to extract the closed concept sets M_i of $\mathcal{M}$. Figure 6 illustrates just two of the resultant concepts. Observe that

```
concept #445:
        events-->       {2, 3, 13, 14, 17, 18, 19, 20, 21, 22, 23}
        generators-->   {{17}, {20}, {21}, {22}, {23}, {2, 18}, {3, 18}, {13, 18},
                         {14, 18}, {2, 19}, {3, 19}, {13, 19}, {14, 19}}
        downpointers--> {446, 1744}

concept #448:
        events-->       {2, 3, 13, 14, 17, 18, 19, 20, 21, 22, 23,
                         50, 53, 54, 55, 58, 59, 61}
        generators-->   {{50}, {53}, {54}, {55}, {58}, {61}, {2, 59}, {3, 59},
                         {13, 59}, {14, 59}, {17, 59}, {20, 59}, {21, 59},
                         {22, 59}, {23, 59}}
        downpointers--> {445, 1743}
```

Fig. 6. Two concepts created by the DDDM system.

our DDDM software also determines the generators of each closed set M_i. Module invocations 17, 20 21, 22 and 23 are each singleton generators of $M_{445} = \{2, 3, 13, 17, 18, 19, 20, 21, 22\}$. That is, {17} logically implies { w, e, 13, 17, 18, 19, 20, 21, 22 }. So too are the conjunctions $2 \wedge 18$, $3 \wedge 18$, ..., $14 \wedge 19$. Clearly the closed sets of a Galois closure are not uniquely generated; it is not antimatroid.

Our DDDM software also enumerates O_{445} and O_{448}, the collections of trace sequences over which these closed sets are formed. Since each is supported by over 1,000 trace sequences, we have omitted this from Figure 6.

As with any closure system, the closed sets M_i of $\mathcal{M}$ (or closed sets O_i of $\mathcal{O}$) form a complete lattice, $\mathcal{L}$, under set inclusion, *c.f.* Figure 1. The "downpointers" of Figure 6 denote the closed concepts covered by M_{445} and M_{448}. For example, we see that M_{448} covers M_{445} in $\mathcal{L}$. When they are small enough, visual inspection of these concept lattices can reveal interesting substructures. Ganter and Wille make full use of this in [2]. The size of our lattice, with 1,805 nodes, precludes visual interpretation. Consequently, we run a search engine on $\mathcal{L}$ to find those closed concepts most likely to be of interest. A prime example are those closed module sets with only a singleton generator, or logical precedent. Concepts #445 and #448 of Figure 6 are examples. In Figure 7, we list all 66 closed concepts denoting implications with at least one singleton precedent.

```
         support singleton
  concept size    generators          closed set
    210  1,034   {1}                -> {1,2,3}
    837  1,160   {2}                -> {2}
    211  1,158   {3}                -> {2,3}
    250    977   {4}                -> {2,3,4}
    836  1,143   {5}                -> {2,5}
    273      3   {6}                -> {1..6}
    118      3   {7}                -> {1..5,7}
     40      3   {8}                -> {1..5,8}
   1789      2   {9}                -> {1..6,8,9}
      1      1   {10,11}            -> {1..11}
   1733  1,099   {12,15,16,24}      -> {2,3,12..24}
    446  1,130   {13,14}            -> {2,3,13,14}
*   445  1,100   {17,20,21..23}     -> {2,3,13,14,17..23}
   1744  1,103   {18,19}            -> {18,19}
    251    966   {25..32}           -> {2,3,5,12..32,56,57}
    389    938   {33}               -> {1..3,5,12..40,56,57}
    392  1,057   {34}               -> {1..3,5,12..32,34..38,56,57}
    391    962   {35,36,37,38}      -> {2,3,5,12..32,35..38,56,57}
    444    975   {39,40}            -> {2,3,13..23,39,40}
    443    977   {41,42,43,44}      -> {2,3,13,14,17..23,40..44}
   1735    863   {45}               -> {2,3,12..24,45}
    945    852   {46}               -> {2,3,5,12..24,46..51,53..55,58..64}
    458  1,077   {47..49,51,60,62} -> {2,3,13..23,47..51,53..55,58..61}
*   448  1,098   {50,53..55,58,61} -> {2,3,13,14,17..23,50,53..55,58,59,61}
    455    960   {52}               -> {2,3,5,13..23,39..44,47..55,58..62}
     53  1,091   {56,57}            -> {2,3,5,12..24,56,57}
   1743  1,101   {59}               -> {18,19,59}
    449    865   {63,64}            -> {2,3,13..23,50,53..55,58,59,61,63,64}
     74    167   {65}               -> {2,3,5,12..24,65}
    375     28   {66}               -> {2,3,5,12..24,39,40,50,53..59,61,66,67,69..72}
    150     96   {67}               -> {2,3,5,12..24,50,53..59,61,67,69..71}
   1754     28   {68}               -> {2,3,5,12..24,50,53..59,61,65,68..72}
    279    100   {69}               -> {2,3,5,12..24,50,53..59,61,69..71}
    452    101   {70}               -> {2,3,5,12..24,50,53..59,61,70}
    581    102   {71}               -> {2,3,5,12..24,50,53..59,61,71}
    583    101   {72}               -> {2,3,5,12..24,50,53..59,61,71,72}
   1745     65   {73,74,75}         -> {18,19,59,73..75}
     37    167   {76}               -> {76}
   1528     39   {77}               -> {1..3,5,12..24,39..44,47..51,53..62,77}
    966    252   {78}               -> {2,3,5,12..24,78,79}
    967    231   {79}               -> {2,3,5,12..24,79}
    358    157   {80}               -> {80}
   1181     61   {81}               -> {1..5,12..45,50,53..59,61,78,79,81}
    232     17   {82}               -> {82}
   1746      7   {83}               -> {18,19,59,73..75,83}
    725      3   {84,117,118}       -> {1..5,12..32,35..45,47..51,53..62,71..75,78,79,83,84,101,114..118}
    833      5   {85}               -> {76,80,85}
    575     62   {86,87,88}         -> {1..5,12..32,35..45,47..51,53..64,76,78..80,82,86..88}
    272      1   {89}               -> {1..6,12..45,47..51,53..65,68..72,76,78..80,82,86..100}
    823      2   {90..100}          -> {2,3,5,12..24,45,47..51,53..65,68..72,76,79,80,90..100}
   1283      3   {95,96,99}         -> {2,3,5,1224,45,47..51,53..62,65,76,79,80,95,96,99}
    731      8   {101}              -> {1..5,12..32,35..40,47..51,53..62,78,79,101}
    439      1   {102,103,105..112} -> {2,3,13..23,39..44,47..55,59..64,70,76,102..112}
   1748      5   {104}              -> {104}
    499      1   {113}              -> {1..5,12..32,35..64,73..76,78..80,82,86..88,113}
   1618      4   {114..116}         -> {1..3,5,12..24,39..44,47..51,53..62,71..75,83,114..116}
```

```
1802      2   {119}           -> {2,3,5,12..24,45..51,53..64,76,80,119}
 764      1   {120}           -> {2,3,5,12..24,45..51,53..65,68..72,76,79,80,90..100,119,120}
1274      1   {121}           -> {1..5,12..45,47..62,65,73..76,78..80,95,06,99,121}
1508      1   {122}           -> {1..3,5,12..24,39..44,46..65,67..77,83,114..116,122}
1676      3   {123..125}      -> {1..3,5,12..24,47..51,53..62,76,78,79,123..126}
1804      4   {126}           -> {1..3,5,12..24,47..51,53..62,76,126}
1742      3   {127..129}      -> {18,19,59,73..75,83,127..129}
1747      4   {130,131}       -> {104,130,131}
1749      4   {132}           -> {132}
1788      1   {133..144}      -> {1..6,8,9,12..64,76,80,82,85,133..144}
```

Fig. 7. Concepts with singleton generators.

Not all of the logical implications in the closure space $\mathcal{I}$ of Figure 7 are interesting; for example #837 is the trivial implication that $2 \rightarrow 2$. Not all of the logical implications correspond to causal dependencies. For example, according to concept #210, $1 \rightarrow \{1, 2, 3\}$, so we know that if module 1 is invoked in any trace then modules 2 and 3 will be also invoked. But invocation of module 1 does not "cause" the invocation of module 2, because in at least one trace the invocation of module 1 is *not* followed by module 2. For $a \rightarrow b$ to be considered a causal dependency, $a \Rightarrow b$, whenever a is invoked, b must be subsequently invoked. In effect, we intersect the closure space $\mathcal{I}$ with the temporal closure space $\mathcal{T}$. Figure 8 illustrates the 41 possible causal dependencies resulting from a single submodule invocation that we found in this set of trace data. A much more complete description of this process can be found in [11].

```
        support
concept size          causal dependencies (possible)

 1733   1,099   {12} => {13...24}
  445   1,100   {17} => {22,23}
                {20} => {21...23}
                {21} => {22,23}
  251     966   {25} => {26} => {27} => {28} => {29}
                {28} => {30} => {31} => {32}          (Note: {29} /=> {30...32})
  391     962   {35} => {36} => {37} => {38}
  444     975   {39} => {40}
  443     977   {41} => {42} => {43} => {44}
  945     852   {46} => {47,48,49,60,62}
  458   1,077   {47} => {48} => {49} => {51} => {60} => {62}
  448   1,098   {50} => {53} => {54} => {55} => {58} => {61}
   53   1,091   {56} => {57}
  449     865   {63} => {64}
  375      28   {66} => {67,69,70}
 1754      28   {68} => {69,70,71,72}
  279     100   {69} => {70}
 1745      65   {73} => {74} => {75}
  966     252   {78} => {79}
  725       3   {84} => {117} => {118}
  575      62   {86} => {87} => {88} => {25...44,45,63,64}
  272       1   {89} => {33,34,90...100)
  823       2   {90} => {91}=>{92}=>{93}=>{94}=>{97}=>{98}=>{100}
 1283       3   {95} => {96} => {99} => {65}
  439       1   {102}=> {103}=>{106}=>{107}=>{108}=>{109}=>{110}=>{111}=>112}
  499       1   {113}=> {46...63,78,79}
 1618       4   {114}=> {115} => {116} => {83}
 1802       2   {119}=> {23...51,55...58,60...64}
  764       1   {120}=> {46...55,58...64,68...72,91...100}
 1274       1   {121}=> {35...45,52,65,95...99}
 1508       1   {122}=> {83}
 1676       3   {123}=> {124} => {125} => {1,2,3,5,12...24,47...51,53...62,78,79}
 1804       4   {126}=> {1,2,3,5,12...24,47...51,53...58,60,61,62)
 1742       3   {127}=> {18,73...75,83,128}
                {128}=> {59,129}
                {129}=> {19}
 1747       4   {130}=> {131}
 1788       1   {133}=> {134}=>{135}=>{136}=>{137}=>{138}=>{139}=>{140}=>{141}=>{142}
                {142}=> {1...6,8,9,12...58,60...64,80,82,119,143,144}
                {143}=> {1...6,8,9,12...58,60...64,80,144}
                {144}=> {1...6,8,9,12...58,60...64}
```

Fig. 8. Possible causal dependencies.

We must point out that the causal dependencies of Figure 8 are what Ernst *et al.* [1, 7] would call "likely" causal events. They have the necessary hallmark of one; the event a is always followed by the event b. But, a scientific assertion of causality based on empirical evidence would also require a plausible explanation of "why a caused b". This is not here. Nevertheless, Yang and Evens, who understand the semantics of this system far better than I, assert that these are true instances of causality.

Figure 8 shows no causal dependencies with conjunctive predicates; yet there are many closed sets with doubleton generators. Concept #445 has eight. Our problem is that, while establishing that b always follows a in a single pass over the trace data is easy, determining that b always follows a conjunction $a_1 \wedge a_2$ is not. But the fact that it is difficult to use the full generality of this intersection model to discover causal relationships does not vitiate the model itself. The fact that any causal dependencies can be found using the model seems to provide strong support for its validity.

4 Transformation of a Causal Topology

The remainder of this paper is more speculative, even though it is supported by a substantial mathematical theory. The issue is to account for change in the causal system with the passage of time. It may well be that $a \wedge b \Rightarrow d$ as in Figure 4, and event a may have occurred. But, if event b does not occur, event d will not occur. And, as the hand of time moves on, the occurrence of a may become irrelevant. What we are trying to say is that Figures 2, 3 and 4 are only static snapshots of the underlying closure systems. In an on-going system, the "topology" of these systems may, or may not, be changing. Thus we are concerned with transformations that take one closure system $(\mathbf{U}, \varphi)$ into another $(\mathbf{U}', \varphi')$. Such a **transformation** $\mathbf{U} \xrightarrow{f} \mathbf{U}'$ is a singled valued function mapping $2^{\mathbf{U}}$ into $2^{\mathbf{U}'}$. By using suffix notation for transformations, including closure operators which also take sets of $2^{\mathbf{U}}$ into sets in $2^{\mathbf{U}'}$, we provide a linguistic indicator of this paradigm shift. Thus $X.f$ denotes the image of the set X under f. We retain the more familiar prefix notation for functions restricted to the underlying set of event elements.

4.1 Regular Transformations

Our goal here is to establish the kinds of properties we would want when transforming a causal topology. By the **inverse transformation** f^{-1} of a set $Y' \subseteq \mathbf{U}'$, we mean the collection $\{Y \subseteq \mathbf{U} | Y.f = Y'\}$.

A transformation $\mathbf{U} \xrightarrow{f} \mathbf{U}'$ is said to be **regular** if it is **union preserving**, that is $(X \cup Y).f = X.f \cup Y.f$. Regular transformations have three important properties that we will use in the following sequence of propositions.

Proposition 3. *If* $\mathbf{U} \xrightarrow{f} \mathbf{U}'$ *is regular, then* f *is:*
monotone, *.e.g.* $\forall X, Y \subseteq \mathbf{U}$, $X \subseteq Y$ *implies* $X.f \subseteq Y.f$;
inverse consistent, *.e.g.* $(\bigcup_Y \{Y : Y.f = Y'\}).f = Y'$
inverse monotone, *.e.g.* $X' \subseteq Y'$ *implies* $X'.f^{-1} \subseteq Y'.f^{-1}$.

Proof. Let $X \subseteq Y$ so $Y = X \cup Y$. Since f is regular, $X.f \cup Y.f = (X \cup Y).f = Y.f$ or $X.f \subseteq Y.f$. Inverse consistency and inverse monotonicity are equally easy to show. □

By inverse consistency we will be able to treat $\bigcup_Y\{Y : Y.f = Y'\}$ as *the* inverse of Y' under f. That is $Y'.f^{-1} = \bigcup_Y\{Y : Y.f = Y'\}$. Inverse consistency also implies that $Y'.f^{-1}.f = Y'$. Thus regular transformations, regarded as operators on a single space, have left inverses, while we have only $Y \subseteq Y.f.f^{-1}$.

Further, it is not hard to show that

Proposition 4. *The composition $f \circ g$ of regular transformations is regular.*

A common way of creating a set valued transformation is to **lift** it from a ordinary point function f on the set $\mathbf{U}$ of events. As usual, we extend f to constituent subsets by defining $Y.f = \{y' \in \mathbf{U}' | \exists y \in Y, y' = f(y)\}$ for all $Y \subseteq \mathbf{U}$. Transformations that are lifted from element functions are easily shown to be union preserving. Thus regular transformations encompass the a class of discrete transformations that we commonly see in practice.

So far we have not considered how a transformation $\mathbf{U} \xrightarrow{f} \mathbf{U}'$ might interact with, or affect, the closures φ and φ' on $\mathbf{U}$ and $\mathbf{U}'$ respectively. A transformation $(\mathbf{U}, \varphi) \xrightarrow{f} (\mathbf{U}', \varphi')$ is said to be **complete** if $\forall Y \in \mathbf{U}$, $Y.f$ closed in $(\mathbf{U}', \varphi')$ implies $Y.\varphi.f = X.f$.

In a discrete space, "completeness" has some of the characteristics more commonly associated with "continuity". In particular, it provides an analog to the notion that "the inverse image of closed sets is closed".

Proposition 5. *Let $(\mathbf{U}, \varphi) \xrightarrow{f} (\mathbf{U}', \varphi')$ be regular and complete. If Y' is closed in $\mathbf{U}'$, then $Y'.f^{-1}$ is closed in $\mathbf{U}$.*

Proof. Let Y' be closed and $Y = Y'.f^{-1}$. Since f is regular $Y'.f^{-1}.f = Y'$ and since f is complete, $Y', f^{-1}.\varphi.f = Y'$, thus $Y'.f^{-1}$ is closed. ☐

Thus the inverse image of a closed set is indeed closed.

Proposition 6. *A regular transformation $(\mathbf{U}, \varphi) \xrightarrow{f} (\mathbf{U}', \varphi')$ is complete if and only if $\forall X \subseteq \mathbf{U}$, $X.\varphi.f \subseteq X.f.\varphi'$.*

Proof. Let $Y' = X.f.\varphi'$, so Y' is closed. Readily, $X.f \subseteq X.f.\varphi'$. Let $Y = Y'.f^{-1}$ Since by inverse monotonicity $X \subseteq X.f.f^{-1} \subseteq Y'.f^{-1} = Y$, by completeness we have $X.\varphi.f \subseteq Y.f = X.f.\varphi'$.
Conversely, let $X.f$ be closed. Then $X.\varphi.f \subseteq X.f.\varphi' = X.f$. Now $X \subseteq X.\varphi$, so $X.f \subseteq X.\varphi.f$ and equality holds. $X.f.\varphi' = X.f$ and f is complete. ☐

We will say a transformation $(\mathbf{U}, \varphi) \xrightarrow{f} (\mathbf{U}', \varphi')$ is **closed** if f takes closed sets in $(\mathbf{U}, \varphi)$ onto closed sets in $(\mathbf{U}', \varphi')$

Proposition 7. *A regular transformation $(\mathbf{U}, \varphi) \xrightarrow{f} (\mathbf{U}', \varphi')$ is closed if and only if $\forall X \subseteq \mathbf{U}$, $X.f.\varphi' \subseteq X.\varphi.f$.*

Proof. Let f be closed. $X \subseteq X.\varphi$ implies $X.f \subseteq X.\varphi.f$. But, because $X.\varphi$ is closed and f is closed, $X.f.\varphi' \subseteq X.\varphi.f$
Conversely, let X be closed in $(\mathbf{U}, \varphi)$. $X.f.\varphi' \subseteq X.\varphi.f = X.f$. But, readily $X.f \subseteq X.f.\varphi'$ so equality holds. ☐

Combining Propositions 7 and 6 we have.

Theorem 1. *A regular transformation* $(\mathbf{U}, \varphi) \xrightarrow{f} (\mathbf{U}', \varphi')$ *is closed and complete if and only if for all* $X \subseteq \mathbf{U}$, $X.\varphi.f = X.f.\varphi'$.

Theorem 1 provides necessary and sufficient conditions on a regular transformation f so that f commutes with the closure operator φ as in Figure 9. In effect, regular transformations that are closed

$$\begin{tikzcd}
\mathbf{U} \arrow[r, "f"] \arrow[d, "\varphi"] & \mathbf{U}' \arrow[d, "\varphi'"] \\
(\mathbf{U}, \varphi) \arrow[r, "f"] & (\mathbf{U}', \varphi')
\end{tikzcd}$$

Fig. 9. Regular f regarded as a closure system transformation

and complete preserve in some fundamental sense the internal closure structure of the underlying discrete spaces. This is just what we want for a slowly evolving causal topology, that is one without catastrophes.

Proposition 8. *Let* $\mathbf{U} \xrightarrow{f} \mathbf{U}', \mathbf{U}' \xrightarrow{g} \mathbf{U}''$ *be regular and complete (closed) then* $\mathbf{U} \xrightarrow{f \circ g} \mathbf{U}''$ *is complete (closed).*

Proof. By Prop. 4, $f \circ g$ is regular. That the composition of closed transformation is closed is trivial. Let f and g be complete and let $Y.(f \circ g) = Y.f.g = Y'' \in \mathbf{U}''$ be closed. We must show that $Y.\varphi.(f \circ g) = Y''$. Since g is complete, $(Y.f.\varphi').g = Y''$ and f complete implies first that $Y.\varphi.f \subseteq Y.f.\varphi'$ and then that $Y.\varphi.f = Y.f.\varphi'$. Thus, $Y.\varphi.f.g = Y.f.\varphi'.g = Y''$. $\square$

Consequently, we can say that the collection of all closure systems[5] whose functors comprise all regular transformations $(\mathbf{U}, \varphi) \xrightarrow{f} (\mathbf{U}', \varphi')$ that are closed and complete constitute a category **ClosureSys**. This category **ClosureSys** has rather nice properties which we will only suggest here. We can define the intersection of transformations $f \cap g$ by $Y.(f \cap g) = Y.f \cap Y.g$ for all $Y \subseteq \mathbf{U}$. It is not at all hard to show that such intersection transformations are themselves regular, closed and complete. Consequently, the collection of all regular, closed, complete morphisms can be equally regarded as an object in **ClosureSys**.

5 Rough Edges

While the idea of modelling causality in terms of an imperative closure system, $\mathcal{I}$, intersected with a temporal closure system, $\mathcal{T}$, may have considerable validity, there are a number outstanding issues, or "rough edges". A first, and foremost, question is whether the kind of causal dependencies illustrated in Figure 8 represent an adequate model, or description, of the underlying deterministic software. The answer is an unequivocal "no". For example, my colleagues, Jinlin Yang and David Evans,

[5] If $|\mathbf{U}| = n > 10$ then there can be more than n^n distinct closure systems defined on $\mathbf{U}$ [8].

model the same software, based on the same traces, in terms of regular expressions that capture the iterative nature of the system much better than Figure 8.

Perhaps the simplest deterministic causal system is a finite state machine. Surely any model of causality should be able to model the behavior of finite state machines for starters. We are familiar with many different ways of describing finite state behavior: regular expressions, graphs, Petri nets, and temporal logics. Each has it strengths and its weaknesses.

Even though the causal dependencies of Figure 8 are not fully adequate to model *JBoss* software, our intersection model can be refined to do a better job. Suppose we let our universe $\mathbf{U}$ of events consist of states s_i, conditions c_j, and actions a_k as suggested in the beginning of Section 2. Now consider the finite state machine M_1 of Figure 10 where $a_1 \equiv (\mathit{read}\ n)$, $a_2 \equiv (n = n-1)$, $a_3 \equiv (\mathit{halt})$

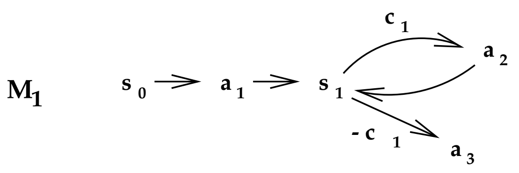

Fig. 10. A simple finite state machine, M_1.

and $c_1 \equiv (n > 0)$. Readily, the set of causal dependencies

$$\begin{aligned} s_0 &\Rightarrow a_1 \\ a_1 &\Rightarrow s_1 \\ s_1 \wedge c_1 &\Rightarrow a_2 \\ a_2 &\Rightarrow s_1 \\ s_1 \wedge \neg c_1 &\Rightarrow a_3 \end{aligned}$$

fully describes M_1. Unfortunately, we do not now know how to adequately instrument software to capture important "conditions" such as $n > 0$ [1, 7], nor how to discover the conjunctive precedents, even though, as shown by Figure 6, the Galois closure of $\mathcal{I}$ is capable of generating the sufficient conjunctive precedents.

An even better description of M_1 in Figure 10 might be "if a number is greater than zero, M_1 decrements it until it is zero". The emphasis here is on the behavior, not on states. To describe behavior, we might first represent the integer n as an ordered set $\mathcal{D}$ of n elements. Now we are using closure to model the data space. We can regard M_1 as a transformation f of $\mathcal{D}$ that simply deletes an element. It is not hard to show that element deletion, in which $\{x\}.f = \emptyset$, and $\{y\}.f = \{y\}, y \neq x$, is regular [10]. (We might note that a deletion transformation cannot be lifted from an ordinary point function because every $x \in \mathbf{U}$ must have an image $f(x) \in \mathbf{U}'$. But, as a transformation $2^{\mathbf{U}} \xrightarrow{f} 2^{\mathbf{U}'}$ it can map $\{x\}$ onto $\emptyset$.) In this case, representing M_1 as a transformation describes "what" it does, not "how" it does it.

It is our sense that the best model of causal behavior may be as a transformation which combines the temporal and imperative aspects of causality. But, unfortunately, the actual formalization of this idea still eludes us.

References

1. Michael D. Ernst, Jake Cockrell, William G. Griswold, and David Notkin. Dynamically Discovering Likely Program Invariants to Support Program Evolution. *IEEE Trans. Software Eng.*, 27(2):1–25, Feb. 2001.
2. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis - Mathematical Foundations.* Springer Verlag, Heidelberg, 1999.
3. Karam Gouda and Mohammed J. Zaki. Efficiently Mining Maximal Frequent Item Sets. In *1st IEEE Intern'l Conf. on Data Mining*, San Jose, CA, Nov. 2001.
4. Robert E. Jamison and John L. Pfaltz. Closure Spaces that are not Uniquely Generated. *Discrete Appl Math.*, 147:69–79, Feb. 2005. also in Ordinal and Symbolic Data Analysis, OSDA 2000, Brussels, Belgium July 2000.
5. Bernard Monjardet. A Use for Frequently Rediscovering a Concept. *Order*, 1:415–416, 1985.
6. Oystein Ore. Galois Connexions. *Trans. of AMS*, 55:493–513, 1944.
7. Jeff H. Perkins and Michael D. Ernst. Efficient Incremental Algorithms for Dynamic Detection of Likely Invariants. *Proc. SIGSOFT'04/FSE-2*, pages 23–32, Nov. 2004.
8. John L. Pfaltz. Evaluating the binary partition function when $N = 2^n$. *Congress Numerantium*, 109:3–12, 1995.
9. John L. Pfaltz. Closure Lattices. *Discrete Mathematics*, 154:217–236, 1996.
10. John L. Pfaltz. Deletion Transformations in Antimatroid Closure Spaces. In *25th Combinatoric, Computing, and Graph Theory Conf.*, Boca Raton, FL, Mar. 1998.
11. John L. Pfaltz. Using Concept Lattices to Uncover Causal Dependencies in Software. In B. Ganter and L. Kwuida, editors, *Proc. Int. Conf. on Formal Concept Analysis, Springer LNAI #3874*, pages 233–247, Dresden, Feb. 2006.
12. John L. Pfaltz and Christopher M. Taylor. Closed Set Mining of Biological Data. In *BIOKDD 2002, 2nd Workshop on Data Mining in Bioinformatics*, pages 43–48, Edmonton, Alberta, July 2002.
13. John L. Pfaltz and Christopher M. Taylor. Concept Lattices as a Scientific Knowledge Discovery Technique. In *Workshop on Discrete Mathematics and Data Mining, 2nd SIAM International Conference on Data Mining*, pages 65–74, Arlington, VA, Apr. 2002.
14. Petko Valtchev, Rokia Missaoui, and Robert Godin. A Framework for Incremental Generation of Frequent Closed Itemsets. In Peter Hammer, editor, *Workshop on Discrete Mathematics & Data Mining, 2nd SIAM Conf. on Data Mining*, pages 75–86, Arlington, VA, April 2002.
15. Rudolf Wille. Restructuring Lattice Theory: An approach based on hierarchies of concepts. In Ivan Rival, editor, *Ordered Sets*, pages 445–470. Reidel, 1982.
16. Jinlin Yang and David Evans. Dynamically Inferring Temporal Properties. In *Proc. ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, PASTE 2004*, Washington, DC, June 2004.
17. Jinlin Yang, David Evans, Deepali Bhardwaj, Thirumalesh Bhat, and Manuvir Das. Terracotta: Mining Temporal API Rules from Imperfect Traces. In *28th Internl. Conf. on Software Engineering (ICSE 2006)*, Shanghai, China, May 2006.