Flows in the Child Welfare Systems: A Computation Theory Approach to Developing Numerical Reference Modes

Peter Hovmand, Melissa Jonson-Reid, and Brett Drake

George Warren Brown School of Social Work, Washington University in St. Louis

ABSTRACT

Service systems are inherently complex, both in their detail and dynamics. System dynamics offers great potential to help policy makers, administrators, and researchers make better decisions about service system changes. However, efforts have been constrained by not being able to construct numerical reference modes without making strong assumptions about the structure of the case flows. This paper presents a novel approach to generating numerical reference modes from administrative databases that is based on computation theory. The method is validated with simulated datasets, and its feasibility and substantive significance demonstrated in an analysis of a merged child welfare database containing 10,250 children and adolescents.

PROBLEM

Administrative databases will contain a variety of information including demographic variables and variables with dates of key events such as the opening or closing of cases. The data may live in a flat database, relational database, or distributed over a variety of unconnected databases. However, most statistical analyses require a matrix or flat database as an input. Thus one will typically construct a single table from multiple tables with each row corresponding to a case and each column representing a variable. In order to capture multiple service dates, one usually creates a series of indicators or date variables for each type of service (for example, see Figure 1).

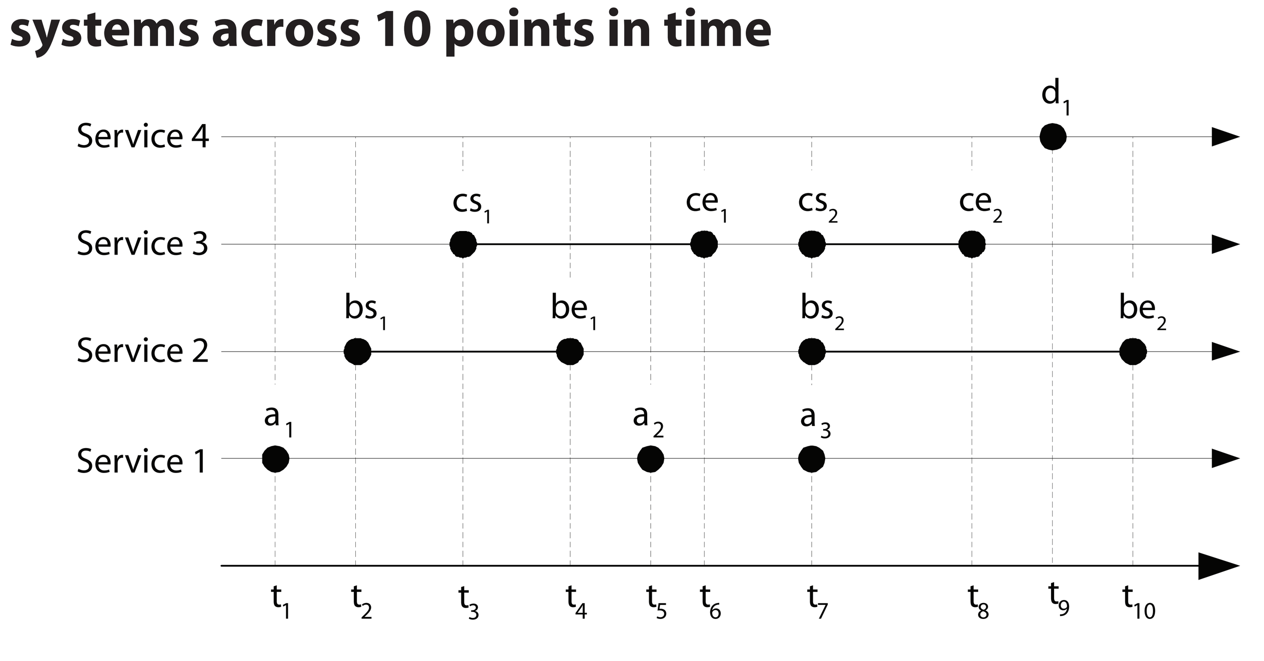
Figure 1: Data matrix with 6 sets of indicators of events and services for 7 individuals

$$
\mathbf{D} = \begin{array}{c|ccccccccccccc}
i & a_1 & a_2 & a_3 & bs_1 & bs_2 & be_1 & be_2 & cs_1 & cs_2 & ce_1 & ce_2 & d_1 & d_2 \\
\hline
1 & 1 & 5 & 7 & 2 & 7 & 4 & 10 & 3 & 7 & 6 & 8 & 9 & . \\
2 & . & . & . & 2 & . & . & . & 2 & 5 & 3 & . & 1 & 4 \\
3 & 1 & . & . & . & . & . & . & . & . & . & . & . & . \\
4 & . & . & . & 1 & . & 2 & . & 1 & . & 3 & . & . & . \\
5 & 2 & . & . & 1 & . & 3 & . & . & . & . & . & 2 & . \\
6 & . & . & . & 1 & . & 2 & . & 2 & . & 3 & . & . & . \\
7 & . & . & . & 2 & . & 3 & . & 1 & . & 2 & . & . & .
\end{array}
$$

Such data are likely to have individuals with co-occurring events and overlapping periods of services (see Figure 2). This presents a problem when there a large number of cases, many types of services and events, and thus uncertainty about the stock and flow structure of a service system. If, however, one had an unbiased algorithm for limiting the number of combinations to a manageable level of detail complexity, then one could build on existing graph or network analysis tools for interactively identifying, exploring, interpreting, and simplifying the major stocks and conserved flows in a service system.

The problem is then to find and specify an algorithm for mapping each row in the data matrix D into a list T of transitions and states such that, T (a) reflects the structure of individuals' pathways through service systems where there are co-occurring events and overlapping periods of service, and (b) is isomorphic with a stock-and-flow description of service pathways.

Figure 2: Individual i=1 passing through four service systems across 10 points in time



METHOD

The main inspiration behind this approach comes from computation theory (e.g., Lewis and Papadimitriou, 1981) in seeing the way that people move through service systems as a finite state machine where the events are letters in an alphabet, the states are words over the alphabet, and the pathways are strings of words or sentences. The solution involves specifying a machine that operates on strings using regular expressions.

The procedure is illustrated in Figures 4 through 5 for the first individual in the data matrix (see Figure 2). The user describes a machine specific to the data set in terms of a state table (Figure 3a) and event table (Figure 3b). The state table maps variables to types of states and events. The event table tells the machine how to recognize events and what operations to perform on the description of the current state. For each individual in the data set, the machine uses the state table to map indicators in the data matrix into a list of events (Figure 4). It then uses the event table to map the list of events into a list of states and transitions (Figure 5).

Figure 3: Description of machine

(a) State table

Name	Regexp	Description
{a}	a.	Service 1
{b}	bs.	Service 2, start
{be}	be.	Service 2, end
{c}	cs.	Service 3, start
{ce}	ce.	Service 3, end
{d}	d.	Service 4

(b) Event table

Event	Lregexp	Rregexp	Description
[{]a[}]	[{]a[}]		Service 1 event
([{]b[}]).*([{]be[}])	([{]b[}])	([{]be[}])	Service 2 interval
([{]c[}]).*([{]ce[}])	([{]c[}])	([{]ce[}])	Service 3 interval
[{]d[}]	[{]d[}]		Service 4 event

Figure 4: Generating a list of events from indicators in the data matrix

$f : \mathbf{D_i} \rightarrow \mathbf{E_i}$

1. Initial list $\mathbf{D_i}$ 2. Map indicators to types of events 3. Sort list of event types by time, t 4. Merge events that occur at the same time into a single event 5. Final list $\mathbf{E_i}$

d	t		e	t	e	t		e	t	e	t
a1	1	→	{a}	1	{a}	1	→	{a}	1	{a}	1
a2	5	→	{a}	5	{b}	2	→	{b}	2	{b}	2
a3	7	→	{a}	7	{c}	3	→	{c}	3	{c}	3
bs1	2	→	{b}	2	{be}	4	→	{be}	4	{be}	4
bs2	7	→	{b}	7	{a}	5	→	{a}	5	{a}	5
be1	4	→	{be}	4	{ce}	6	→	{ce}	6	{ce}	6
be2	10	→	{be}	10	{a}	7				{a}{b}{c}	7
cs1	3	→	{c}	3	{b}	7	→	{a}{b}{c}	7	{ce}	8
cs2	7	→	{c}	7	{c}	7				{d}	9
ce1	6	→	{ce}	6	{ce}	8	→	{ce}	8	{be}	10
ce2	8	→	{ce}	8	{d}	9	→	{d}	9		
d1	9	→	{d}	9	{be}	10	→	{be}	10		
d2	.										

Figure 5: Generating a list of states and transitions from a list of events

$g : \mathbf{E_i} \rightarrow \mathbf{T_i}$

6. Initial input list $\mathbf{E_i}$

7. The machine reads each line **j** of input from $\mathbf{E_i}$ and adds that to the last machine state, $\mathbf{Q_{k-1}}$, to generate a description of the individual's current state, $\mathbf{Q_{k-1} + e}$. The machine then applies the regular expressions from the event table to $\mathbf{Q_{k-1} + e}$, which becomes the next machine state, $\mathbf{Q_{k+1}}$.

8. Final output list $\mathbf{T_i}$

j = 1 ▶

e	t
{a}	1
{b}	2
{c}	3
{be}	4
{a}	5
{ce}	6
{a}{b}{c}	7
{ce}	8
{d}	9
{be}	10

	Input at j			Machine states		
j	e	t	k	Q_{k-1}	Q_{k-1} + e	Q_{k+1}
1	{a}	1	1	{Unk_left}	{Unk_left}{a}	{a}
			2	{a}	{a}	{Unk_a}
2	{b}	2	3	{Unk_a}	{Unk_a}{b}	{b}
3	{c}	3	4	{b}	{b}{c}	{b}{c}
4	{be}	4	5	{b}{c}	{b}{c}{be}	{c}
5	{a}	5	6	{c}	{c}{a}	{a}{c}
			7	{a}{c}	{a}{c}	{c}
6	{ce}	6	8	{c}	{c}{ce}	{Unk_c}
7	{a}{b}{c}	7	9	{Unk_c}	{Unk_c}{a}{b}{c}	{a}{b}{c}
			10	{a}{b}{c}	{a}{b}{c}	{b}{c}
8	{ce}	8	11	{b}{c}	{b}{c}{ce}	{b}
9	{d}	9	12	{b}	{b}{b}	{b}{d}
			13	{b}{d}	{b}{d}	{b}
10	{be}	10	14	{b}	{b}{be}	{Unk_b}
			15	{Unk_b}	{Unk_b}	{Unk_right}

S_k	S_{k+1}	t	Imputed
{Unk_left}	{a}	1.0	F
{a}	{Unk_a}	1.1	T
{Unk_a}	{b}	2.0	F
{b}	{b}{c}	3.0	F
{b}{c}	{c}	4.0	F
{c}	{a}{c}	5.0	F
{a}{c}	{c}	5.1	T
{c}	{Unk_c}	6.0	F
{Unk_c}	{a}{b}{c}	7.0	F
{a}{b}{c}	{b}{c}	7.1	T
{b}{c}	{b}	8.0	F
{b}	{b}{d}	9.0	F
{b}{d}	{b}	9.1	T
{b}	{Unk_b}	10.0	F
{Unk_b}	{Unk_right}	-1.0	T

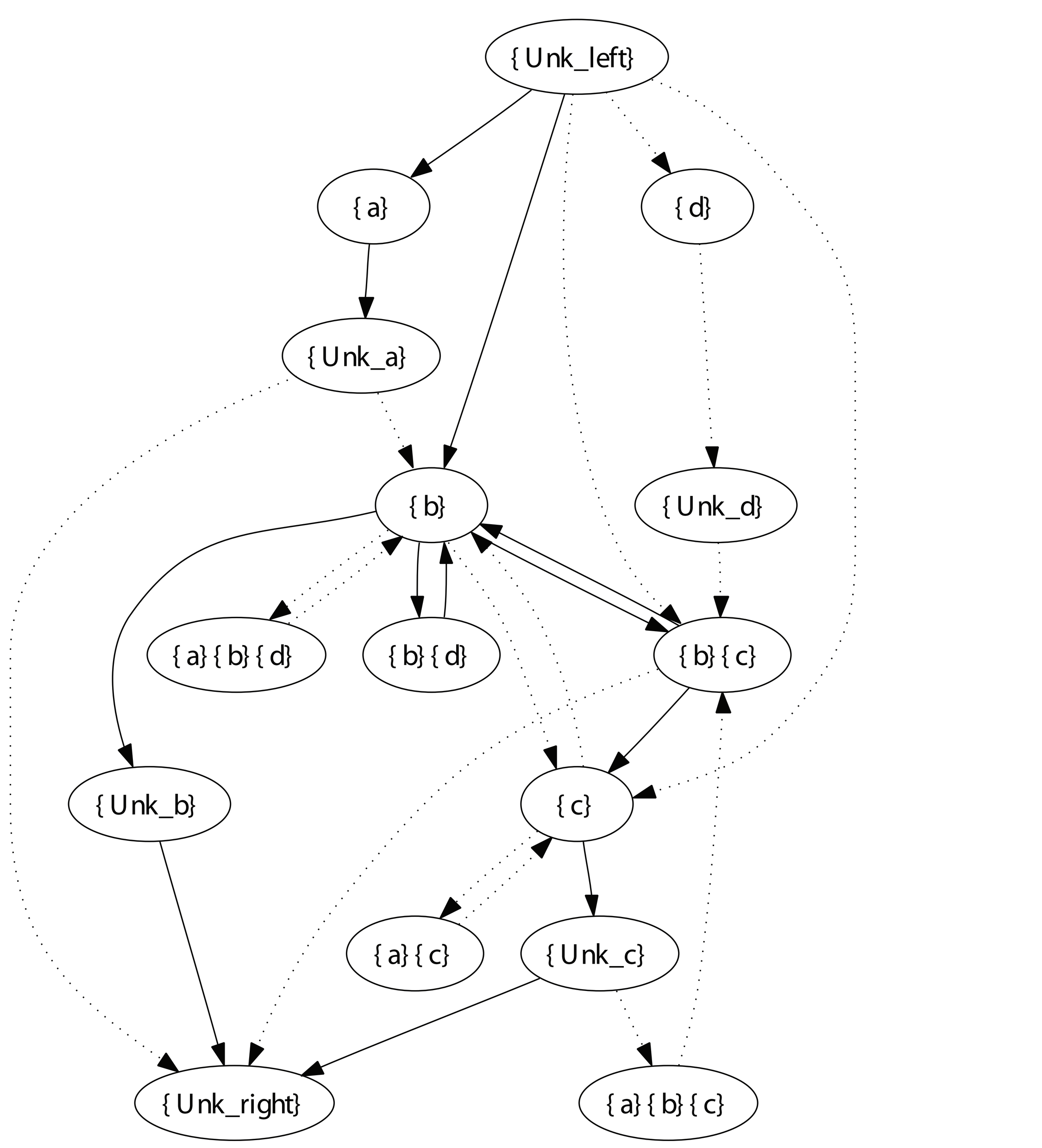
RESULTS

The output is a list of states, transitions, and times that can be used for a wide range of analyses including the graph and network analysis, calculation of transition matrices, numerical reference modes, and estimation of delays. Figure 6 is a graph of the network generated from the data shown in Figure 1 and isomorphic with a stock and flow diagram of the pathways through a service system.

The circles represent states or stocks, while the arcs represent case flows with the arrow pointing in the direction of the flow. Solid lines indicate where two or more individuals passed through a flow, while dashed lines indicate that only one person passed through the flow.

The procedure is practical for relatively large merged administrative databases. A merged database with 10,250 individuals and 27 types of states and events over nearly 200 indicator variables required two hours to process on a Dell Pentium 4 with 512 MB of RAM. The computation time required is linear, and processing can easily be distributed over several computers and run incrementally as additional data become available

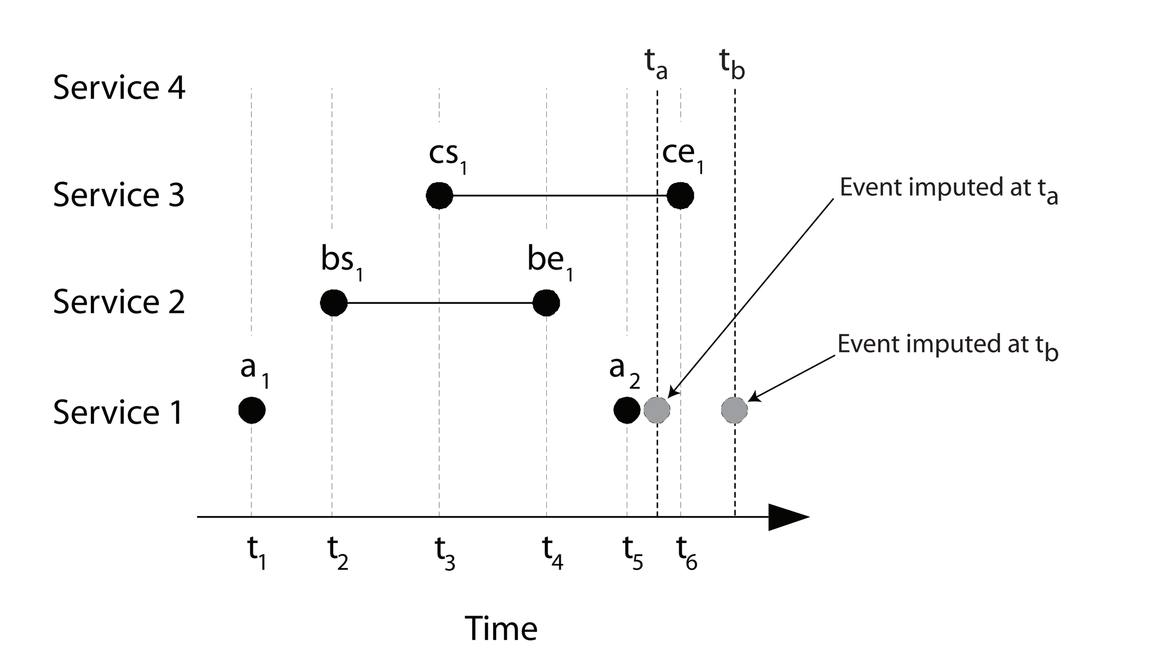
Figure 6: Graph of network flows



LIMITATION

Imputing transition events introduces an uncertainty in some states. For example, in Figure 7, imputing the transition at t_a will result in a transition from {a}{c} into {c}, while imputing the transition at t_b will result in a transition from {a} into {unk_a}. This could lead to descriptions of the service network and reference modes that are sensitive to assumptions about the delay between {a} and the imputed event. This is an inherent limitation of trying to map a hybrid representation of a system into a continuous representation.

Figure 7: Imputation of unknown states and times and



CONCLUSIONS AND NEXT STEPS

The major benefit of this approach is that it facilitates the exploration and reduction of the stock and flow structure of a service network that has high detail complexity. It can be implemented in a variety of programming languages and is relatively efficient, and thus practical to use on large data sets. Corresponding numerical reference modes and estimates can automatically be extracted from the reduced network, and recalculated as one gains a deeper understanding of the system.

Next steps include (1) analyzing the child welfare network with respect to a variety of outcomes, and identifying the major pathways and feedback loops contributing to those outcomes, and (2) comparing service pathways by demographics and risk groups. Working with stakeholders and other researchers, these results will then be used as the basis for a system dynamics model of a dynamic problem in the child welfare service system.