¡s¿ C aus ality and the Sem antics of Pro ven ance James Cheney ar X iv : 100 4 . 3 241 v 1 [cs . PL] 19 Apr 2010 L FC S University of Edinburgh j che ney @ inf . ed . ac . uk Pro ven ance , or information about the sources , deriv ation , custody or history of data , has been studied recently in a number of contexts , including databases , scientific work flows and the Sem antic Web . Many proven ance mechanisms have been developed , motivated by informal notions such as influence , dependence , explanation and caus ality . However , there has been little study of whether these mechanisms formally satisfy appropriate policies or even how to formal ize relevant motivating concepts such as caus ality . We contend that mathematical models of these concepts are needed to justify and compare proven ance techniques . In this paper we review a theory of caus ality based on structural models that has been developed in artificial intelligence , and describe work in progress on a causal semantics for proven ance graphs . 1 Introduction Pro ven ance is a general term referring to the origin , history , chain of custody , deriv ation or process that yielded an object . In analog settings such as art and archae ology , such information is essential for understanding whether an artifact is authentic , valuable or meaningful . In the digital world , proven ance is now recognized as an important problem because it is very easy to silently alter or forge digital information . We already pay a price because of the lack of robust mechanisms for recording and managing proven ance : serious economic losses have been incurred due to the lack of proven ance on the Web [3], and lack of transparency of scientific processes and results is routinely used to sow confusion and doubt about climate change [25]. Much work on proven ance considers the following basic scenario : we have some input data and some complex process that will be run on the input , for example a large program , possibly split into many smaller jobs and executed in parallel or on a distributed system . In this setting , a proposed solution generally records some additional information as the program runs . The additional information is often called proven ance and it is supposed to provide an ex plan ation showing how the results were obtained . Of course , this is a loose specification if we do not clarify what we mean by an explanation (ap art from whatever proven ance information happens to be recorded by the system). There are at least two obvious - se eming choices , neither of which seems satisfactory in practice : First , we might record the program that was run along with its input data (and any intermediate inputs such as user or network interactions). This , at least , allows us to re run the program later and check that we get the same result , and it also allows us to vary the inputs to see how changes affect the output . But this is nearly useless as an explanation , especially for end - users who are not (and should not be expected to be) proficient at debugging black - box systems . Moreover , the inputs and outputs may be huge (for example , gig abytes of climate data), and it may not be possible for users to manually inspect the data . In the longer term , just recording the program is also problematic since the computational environment in which the program runs will change in some sense , this is also an input that we cannot feas ibly record . Sub mitted to : 2 Second , we might record everything that can be recorded about the computation , in the hope that it might someday be useful . This also sounds straightforward but is surprisingly difficult to pin down , since everything that can be recorded can be interpreted in many different ways . Should we record every function call ? Every instruction ? Every molecular interaction ? Do we need to record what the programmer had for breakfast on the day the program was written ? Clearly we have to stop somewhere , and for efficiency reasons we should probably stop far short of any reasonable definition of everything . Most extant approaches pick some intermediate point between these two extremes , committing to some (often not explicitly stated) choices about what is important about the computation that should be recorded in its proven ance . For example , in databases , there are models such as where - proven ance [2] (tracking the s ources of copied data), lineage [10], why - proven ance [2] or how - proven ance [14] (tracking tu ples used by or that just ify a result tuple), or dependency proven ance [7] (a comput able approximation of the information flow behavior of the program). Pro ven ance has also been studied extensively in other settings , particularly scientific workflow systems (e . g . [21 , 20 , 18]). Scientific work flows are usually high - level , visual programming languages , often based on data flow or Pet ri - net models of concurrent computation , and often executed on grid or cloud computing platforms . This architecture has the advantage that it puts considerable computational power into the hands of scientists without forcing them to learn how to program parallel or distributed systems at a low level in C ++ or Java . However , it also has a serious drawback : distributing a program over a heter ogeneous network dramatically increases the number of things that can go wrong , typically makes the computation nond etermin stic and makes it hard for the user to trust the results . Scientists are reluctant to publish results based on programs that may contain subtle bugs , and whose behavior is different every time they are run , or depends on libraries or other environmental factors in subtle ways . Pro ven ance is perceived as important for helping users understand whether results of such comput ations are repeat able and trustworthy , and in particular for scientists to be able to judge the scientific validity of results they may wish to publish . The work on database proven ance is distinctive in that several different formal models have now been defined for database query languages with well - under stood semantics . This makes it easier to compare , relate and general ize these approaches , though such comparisons are only starting to appear [8 , 14]. For most of these models , there are semantic guarantees (or even exact semantic character izations) relating the proven ance records to the den otation of the program . On the other hand , for workflow proven ance , formal definitions of the meaning of workflow programs have only started to appear recently (see for example [29 , 17]), while the proven ance semantics of these tools is usually specified inform ally , at best [21]. As a result there is a confusing variety of models and styles of proven ance for work flows . To address this problem , there has been an ongoing community effort , centered on a series of Pro ven ance Challenges [23], to understand and compare the qualitative behavior of these different systems and synthes ize a common format for exchanging proven ance among them . This effort has recently yielded a draft Open Pro ven ance Model , or O PM [22]. Inst ances of this model are graphs whose nodes represent agents , processes or artifacts and whose edges represent dependence , generation or control relationships . The O PM has sem antics in the sense of the Sem antic Web , in that the nodes and edges are expected to have names that are meaningful to reasonably well - informed users . The O PM standard draws heavily on informal motivations such as proven ance is the process that led to a result and ed ges denote causal relationships linking the cause to the effect . But while the O PM specifies a graph notation , controlled vocabulary for the edges , and inference rules for infer ring new edges from existing edges , it J . Cheney 3 does not have a sem antics in the den ot ational or operational sense by which we might judge whether a graph is consistent or complete or whether inf erences on the graph are valid . In this paper , we investigate the use of structural causal models [24] as a semantics for these graphs , and relate the informal motivations invoked in defining O PM graphs with the formal definitions of actual cause and explanation due to Hal per n and Pearl [15 , 16]. We do not argue that structural causal models provide the only or best causal account of proven ance . However , structural causal models are quite close to O PM - style proven ance graphs (mod ulo cosmetic differences), so the analogy is compelling . Moreover , structural models have been studied extensively (e . g . [24 , 15 , 16 , 11 , 12 , 13]) and have proven useful to both philosophical accounts of scientific explanation [30] and psychological theories of understanding [27]. Nevertheless , it may be enlight ening to apply other mathematical theories of caus ality and explanation to proven ance , or investigate variations and extensions of Hal per n and Pearl s approach . The broader aim of this paper is to argue by example that semantics (in the mathematical sense) is badly needed for research on proven ance . One of the major motivations for proven ance is to improve scientific communication , by allowing scientists to generate and exchange computational ex plan ations or just ifications of their results . In biology , for example , some journals now require both data and workflow programs describing how results were obtained , and some scientists anticipate that scientific publication will evolve into richer documents incorporating text , data , and computation [26]. However , if the techniques used to do so are poorly specified and un verified then we can expect errors and confusion . Programming languages and semantics researchers can and should be involved in making sure that these techniques are clearly described and robust , to help ensure that scientific communications retain long term value as they gain computational structure . 2 Examples Before del ving into technical details , we give a high - level example comparing O PM - style proven ance graphs with structural causal models . The left - hand side of Figure 1 shows a simple O PM graph , based on a standard example showing the proven ance of a cake [22]. The right - hand side shows a structural causal model , depicted as a graph . These two graphs are intentionally very similar . In the O PM graph , the o vals denote artifacts (¡/s¿