Templates as a method for implementing data provenance in decision support systems

Vasa Curcin[a,*], Elliot Fairweather[a], Roxana Danger[b], Derek Corrigan[c]

[a]*Division of Health and Social Care Research, King's College London, London, United Kingdom*
[b]*Reed Online Ltd, London, United Kingdom*
[c]*Royal College of Surgeons in Ireland, Dublin, Ireland*

Abstract

Decision support systems are used as a method of promoting consistent guideline-based diagnosis supporting clinical reasoning at point of care. However, despite the availability of numerous commercial products, the wider acceptance of these systems has been hampered by concerns about diagnostic performance and a perceived lack of transparency in the process of generating clinical recommendations. This resonates with the Learning Health System paradigm that promotes data-driven medicine relying on routine data capture and transformation, which also stresses the need for trust in an evidence-based system. Data provenance is a way of automatically capturing the trace of a research task and its resulting data, thereby facilitating trust and the principles of reproducible research. While computational do-

*Corresponding author
Email addresses: vasa.curcin@kcl.ac.uk (Vasa Curcin), elliot.fairweather@kcl.ac.uk (Elliot Fairweather), roxana.danger@reedonline.co.uk (Roxana Danger), derekcorrigan@rcsi.ie (Derek Corrigan)

mains have started to embrace this technology through provenance-enabled execution middlewares, traditionally non-computational disciplines, such as medical research, that do not rely on a single software platform, are still struggling with its adoption. In order to address these issues, we introduce provenance templates – abstract provenance fragments representing meaningful domain actions. Templates can be used to generate a model-driven service interface for domain software tools to routinely capture the provenance of their data and tasks. This paper specifies the requirements for a Decision Support tool based on the Learning Health System, introduces the theoretical model for provenance templates and demonstrates the resulting architecture. Our methods were tested and validated on the provenance infrastructure for a Diagnostic Decision Support System that was developed as part of the EU FP7 TRANSFoRm project.
Keywords: D2.1 (Software Engineering) Requirements/specification J.3 (Life and Medical Sciences): Health
data provenance, model-driven architectures, decision support systems

1. Introduction

The importance of data, its origins and quality, has long been recognised in clinical research. In recent years, we have also witnessed increased reliance of clinical practice on data, through routine data capture in Electronic Health Record systems, quality improvement initiatives at multiple levels, and growing adoption of evidence-based medicine.

The patient safety implications of diagnostic error in family practice are potentially severe for both patient and clinician [1]. The development of

diagnostic clinical decision support systems (DSS) has long been advocated to promote consistent guideline-based diagnosis supporting clinical reasoning at point of care. However, the wider acceptance of these systems in clinical practice has been much slower in happening despite the availability of many commercial products. Concerns remain about diagnostic performance and a perceived lack of transparency in first generation systems that deploy an evidence knowledge base in the form of a black box that generates clinical recommendations. These concerns about the quality of evidence and the effort required in the longer term maintenance and sustainability of the underlying evidence base supporting such systems has lead to research into second generation tools supporting a more dynamic and iterative cycle of evidence creation and update using a technical infrastructure developed under the auspice of the Learning Health System (LHS) [2].

The Learning Health System community envisages every participant in the health system (clinician, patient, researcher, insurer...) as both a producer and consumer of data. Central to this vision is the notion of routine capture, transformation and dissemination of both data and resulting knowledge. Clinical studies, quality improvement initiatives, decision support, and other scenarios can all then be associated with the routes that the data is taking through the LHS. The trust information associated with the data needs to be made available at each step of these use cases, to support auditability and transparency.

When applied to DSS-s, this trust requirement translates to the ability to readily demonstrate the clinical reasoning that was performed in a clinical encounter, together with the recommendation received. In addition

to supporting the auditability of the process, this capability also promotes transparency and traceability from the recommendation back to the rules applied to produce the recommendation.The data provenance community has been working on methods for ensuring reproducibility in scientific research, through use of Semantic Web techniques and the W3C PROV standard [3], that are highly relevant to the challenges of decision support in the LHS environment. Computational provenance provides a uniform data-centered audit trail of what actually happened during some task, and we shall describe how these methods can be adapted to the needs of LHS.

There are two main techical challenges to be addressed in applying data provenance to the Decision Support System scenario; firstly, how to have heterogeneous, distributed software agents (security systems, rule engines...) construct unified, verifiable provenance traces, and secondly, how to formally guarantee that the resulting provenance traces will satisfy domain constraints, often expressed in ontologies, and user data requirements.

In order to address these issues, we introduce *provenance templates*, abstract provenance fragments representing meaningful domain actions that can be used to generate a model-driven service interface for domain software tools to routinely capture the provenance of their data and tasks. A template defines a provenance graph in a generic manner by means of variables such that it may be later instantiated and grafted onto pre-existing provenance graphs. Importantly, this paper introduces the idea that templates may describe subgraphs subject to bounded iteration in both serial and parallel manner.

The EU FP7 TRANSFoRm project [4] has developed a diagnostic deci-

sion support tool that promotes numerous state-of-the-art practices of good clinical decision support. These include precisely defined usability patterns, integration with an electronic health record (EHR), allowing for recommendations at the point of care as part of the clinician workflow, and a provenance backend that captures provenance data about the computational aspects of the diagnostic task.

The paper first introduces the concepts of the Learning Health System, data provenance and decision support systems in section 2, before presenting the requirements of the LHS-enabled DSS, novel provenance templates formalism and the associated provenance architecture in section 3. Section 4 demonstrates how the new model was used to construct DSS audit trails in TRANSFoRm and in section 5 we consider how our approach addresses the wider LHS requirements for trust in decision support systems, its impact with respect to some recent developments, and list related work. Section 6 offers conclusions and presents pointers for future research.

2. Background

We shall now review the Learning Health System paradigm and the data provenance technologies, and relate them to the challenges of clinical Decision Support Systems, presenting as an example the DSS developed as part of the TRANSFoRm project.

2.1. Learning Health System

The Learning Health System (LHS) movement aims to establish a next-generation healthcare system, "... one in which progress in science, informatics, and care culture align to generate new knowledge as an ongoing,

natural by-product of the care experience, and seamlessly refine and deliver best practices for continuous improvement in health and health care." [5] Each participant in the LHS, be they clinician, patient, or researcher, acts as a consumer and a producer of knowledge, with the LHS providing: a) routine and secure aggregation of data from multiple sources, b) conversion of data to knowledge and c) dissemination of that knowledge, in actionable form, to everyone who can benefit from it [2]. Thus, the LHS creates routes for knowledge transfer between different parts of the health system, thereby increasing its research and learning capacity.

Different data-driven scenarios, such as decision support systems, clinical trial recruitment and management, epidemiological studies, all represent applications within the LHS, each associated with the movements and processing of data and knowledge. A number of LHS implementations have been developed at varying scales [4, 6, 7, 8].

Attempts to define the core requirements of the Learning Health System [5] have highlighted concerns about a perceived lack of transparency and tracking in current systems demonstrating how clinical reasoning was actually applied in any given clinical case. A fundamental feature of the LHS is the generation and curation of clinical evidence using electronic data sources. Such a process is critically dependent on a full transparency of how evidence is produced, maintained and consumed as a means of generating trust in the underlying system. Trust in the evidence base leads to the acceptance of responsibility for the clinical recommendations made by it which is essential if these tools are to gain widespread acceptance in the clinical community.

2.2. Data provenance

Put simply, data provenance describes what actually happened for some data entity to achieve its current form. W3C standards body defines provenance as a form of contextual resource metadata *that describes entities and processes involved in producing and delivering or otherwise influencing that resource. Provenance provides a critical foundation for assessing authenticity, enabling trust, and allowing reproducibility.* The Office of the National Coordinator (ONC) for Health IT describes it as *attributes about the origin of health information at the time it is first created and tracks the uses and permutations of the health information over its lifecycle.* Term *data provenance* is used to establish the focus on data entities produced in the processes.

Data provenance provides traceability by automatically capturing the trace of the research task and resulting data in a uniform and domain-indepent way, thereby facilitating reproducible research. The original concept comes from the eScience and cyber-infrastructure communities, where it was used for capturing the exact parameterisations and configurations of scientific workflows that produced a particular data set [9, 10]. Although the original users of provenance data were the scientific programmers creating and maintaining research workflows, the increasing number of tools and technologies available resulted in a wide array of stakeholders who can benefit from provenance information using visual front-end tools and interactive reports.

2.2.1. PROV model

The provenance technology, as defined in the W3C PROV standard[3], provides a common platform for automated capture of metadata about the

data artifacts (e.g. databases, individual patient records, diagnostic recommendations), all processes that use or create those artifacts, and all actors that participate in those processes, such as clinicians, patients, researchers, or computer software. The resulting provenance data stores are typically semantically annotated databases, shared between all different software tools in some software system that can be mined for generating new knowledge, or investigated for audit purposes [11].

PROV is an interoperability standard, so there is no need for every system to use it as its core data model, or even to use a graph data model, but the W3C recommendation is for each provenance-enabled system to support import and export in the PROV format.

Nodes in a provenance graph come in three flavours: **entities**, which represent immutable states of a some data for which one wants to provide a history, **activities** that produce and consume such entities and **agents** associated in some capacity with either of the former. The **edges** of a graph represent various inter-relations between the node types, such as usage, generation, and association [3]. Validity of graphs is defined using a number of typing, ordering and impossibility constraints to be checked upon a normalised form of a graph, if one exists [12]. All nodes have a mandatory **identifier** given as a **qualified name**. A qualified name consists of an optional **namespace** followed by a local name of form **ns:name**. Identifiers belong to the `prov` namespace. Nodes and edges may be annotated with an optional dictionary of **attribute-value pairs**, formed of a qualified name and a data value, which can be used to attach ontological annotations onto nodes, specifying their meaning in some domain. Fig. 1 demonstrates these

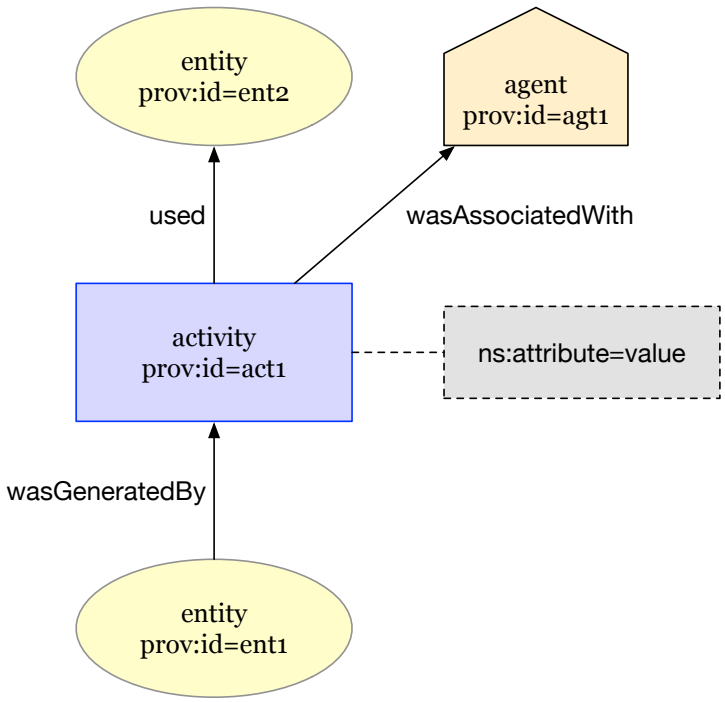

Figure 1: An example PROV graph using the standard representation of entities as yellow ellipses, activities as blue rectangles and agents as orange pentagons. Node annotations are shown as dashed grey boxes.

features in diagrammatic form using the standard PROV representation of entities as yellow ellipses, activities as blue rectangles and agents as orange pentagons. Node annotations are shown as dashed grey boxes.

2.3. Clinical Decision Support Systems

Decision support systems (DSS) have a long and sometimes controversial research history [13, 14]. Clinical decision support system is defined as software that is designed to be a direct aid to clinical decision-making, in which the characteristics of an individual patient are matched to a computerized

clinical knowledge base and patient-specific assessments or recommendations are then presented to the clinician or the patient for a decision [15].

The exact nature of the patient-specific assessments or recommendations and the delivery mechanism used to present that information to the patient or clinician can vary greatly [16]. This has resulted in a number of different types of clinical decision support system that address particular clinical areas, ranging from computerised physician order entry and appropriate medicines management, via risk calculators, diagnostic aids, and triggered alerts and reminders to full electronic implementations of clinical guidelines.

The demonstrable efficacy of DSS in clinical practice however has been limited. One reason is that research impacts of implementing such systems have frequently been assessed as a technical driver of process change. Ideally they should more usefully demonstrate a measurable positive impact on practitioner performance that leads to directly attributable and measurable improvement in patient outcomes [17]. But more promising results have been demonstrated in research environments outside the clinical area of diagnostics [18, 19, 20, 21].

Traditional approaches to diagnostic decision support have lacked broad acceptance for a number of other well documented reasons: poor integration with EHRs and clinician workflow, static black-box rule based evidence that lacks transparency and trust, usage of proprietary technical standards hindering wider interoperability [22, 18, 23, 24, 25]. Despite these problems there is an increasing recognition of the need to realise the potential value of implementing decision support systems more generally. This is reflected in their inclusion as important components of wider government ICT based

health policy legislation in practice [26, 27].

The evolution of clinical decision support development reflects attempts to address workflow and integration issues, interoperability standards and also separation of the knowledgebase as a separate service distinct from the tools themselves [28]. The focus has largely been on implementations of what can be described as diagnostic symptom checkers, relying on a knowledge base defined as a series of rules in the form of a database of knowledge facts. These may be triggered or combined together in the form of guidelines based on statements using a knowledge rule languages or rule engines such as Arden Syntax [29], GLIF [30] and GELLO [31]. These approaches have led to a recent shift towards model-based approaches to knowledge representation for the purposes of clinical decision support[32].

2.4. TRANSFoRm Decision Support System

The EU FP7 TRANSFoRm project (2010-2015) [4], working with 20 partners in 10 European countries, developed and evaluated a single unified international platform to support main Learning Health System scenarios that combine research and clinical practice, and reduce barriers to entry for using Electronic Health Record (EHR) systems and large medical data sources. The project developed a next generation diagnostic decision support tool that addresses many of the issues highlighted as being essential for good clinical decision support [22]. These include integration with an electronic health record (EHR) allowing for recommendations at the point of care as part of the clinician workflow. An essential part that is the subject of this research paper has been the support for the LHS concepts of transparent generation and use of evidence in this system.

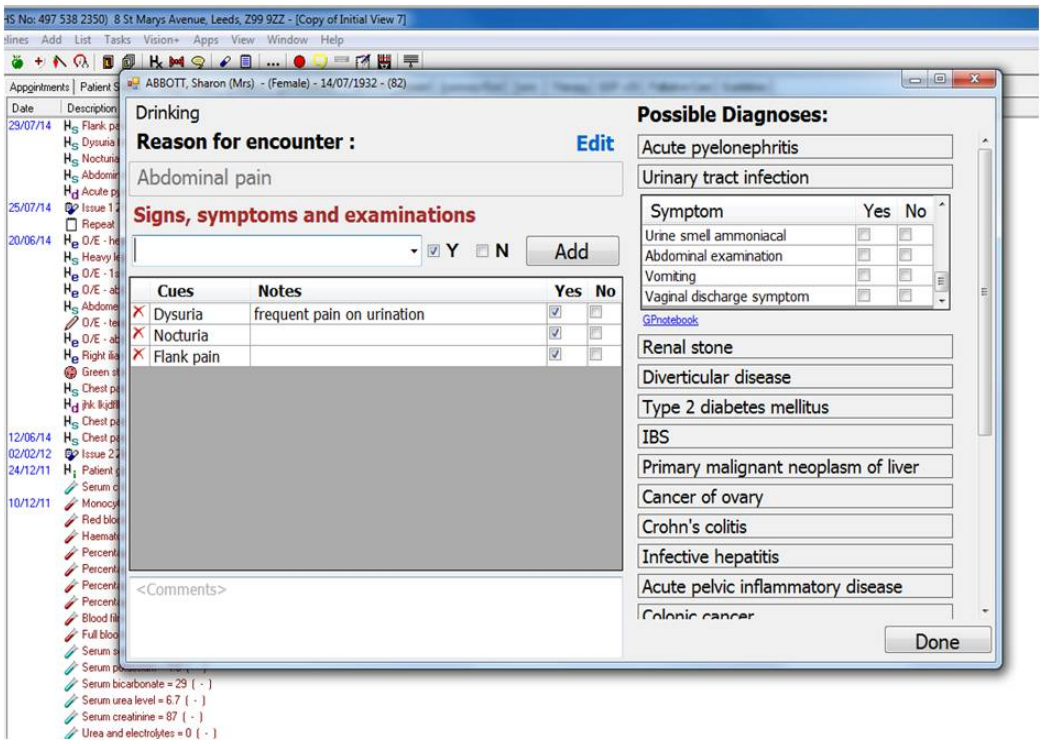

Figure 2: TRANSFoRm Diagnostic decision support tool

A prototype next generation diagnostic decision support system was developed in TRANSFoRm as part of a wider learning health system infrastructure. The tool, shown in Figure 2 is driven by clinical knowledge obtained through a web service based clinical evidence repository providing model driven prompting and recording of coded patient diagnostic cues supporting diagnosis of 78 clinical conditions. The diagnostic decision support tool is embedded and interoperable with the workflow of an EHR system in family practice (Vision 3 EHR [1]) as shown in Figure 2. The tool allows for bottom-up input of observed patient cues (left-hand window) or top-down drilling into and selection of evidence cues supporting a specific diagnosis for investigation (right-hand window). A dynamically updated cue count is maintained for each differential diagnosis indicating the number of evidence cues observed as present for each diagnosis based on the patient cues recorded during the consultation. This allows dynamic ranking of potential differential diagnoses being considered (the most likely at the top) based on the patient presenting reason for encounter, along with a record of the evidence supporting each diagnosis under consideration. Upon exiting the tool a working diagnosis can be confirmed and the coded evidence cues and current working diagnosis can be saved back and recorded for future reference in the patient EHR. A diagnostic evidence ontology [33] was created to serve as the central information model for the DSS tasks, supporting provision of diagnostic evidence for over 70 diagnostic conditions to decision support consumers using a web service interface. The evidence content can either be manually

[1] `www.inps.co.uk/vision`

curated or populated using a separately developed data mining module [34]. The DSS system used the provenance infrastructure that forms a core part of the TRANSFoRm middleware, together with the security framework and semantic interoperability modules.

3. Material and methods

We shall now look into how TRANSFoRm implemented the provenance infrastructure for its diagnostic decision support system. First, we shall present the requirements stemming from the context of the Learning Health System, and then present the theoretical framework for provenance template architecture.

3.1. Reproducibility requirements of a provenance-enabled decision support system

To inform our design for provenance templates as means of implementing reproducibility in DSS, we now establish the reproducibility requirements for a provenance-enabled DSS, by placing them in the context of the key Learning Health System challenges [5]:

- *An LHS that is trusted and valued by the public and all stakeholders.* Privacy, security, and transparency are key elements related to building public trust and generating value. Trust and confidence at all stages of the LHS operation are essential; from inputs to outputs (and outcomes). This implies the need for **traceability** - a continuous trail of data artifacts and operations on those artifacts, starting from the data creation (e.g. routine data capture or import from a data source)

through the transformations (knowledge base processing, rule application) all the way to recommendations made by the DSS.

- *An Adaptable, Self-improving, Stable, Certifiable, and Responsive LHS.* In the context of an adaptable system, how do we determine what adapts? How can a system adaptably ingest, manage, refine, and emit data from a rapidly growing source environment? What evidence must be gathered about the development, design, and operation of the system and about the environment in which it operates to enable certification? The LHS software architectures need to provide a mechanism for such evidence to be **routinelly curated** - gathered, organized, interpreted, and maintained.

- *An LHS Capable of Engendering a Virtuous Cycle of Health Improvement.* How do we develop ways to communicate the generated results, information, or knowledge to others who may wish to replicate (or build upon) the work done, as well as to the general public? How can the computational procedures employed in the system be documented in ways that are assuredly consistent, understandable, checkable, and repeatable, and how can the computational provenance of derived data be tracked from its points of production through consumption and use? These features rely on permanent **auditability** of the system, with all neccessary audit data being automatically generated from the provenance traces, and the models

Based on these, we define the key reproducibility requirements that apply to decision support.

1. **System transparency.** The black box approach and lack of transparency results in the lack of trust and is cited as one of the main reasons behind the poor take-up of clinical decision support systems [35]. Therefore, in a provenance-enabled DSS, activities related to usage and generation of evidence need to be readily available for users to review.
2. **Auditability of recommendations.** Medical/legal liability concerns are considered a potential stumbling block for Decision Support Systems [22], in that it is unclear who takes responsibility for various elements in the DSS that could potentially go wrong. This relates to the **auditability** of the system, which must enable the user to look up a diagnostic recommendation and find all the relevant detail about how it was made - evidence base used, patient cues entered, software employed. The level of detail captured must be validated against the required report granularity.
3. **Understandability of data.** The data that is captured about the workings of the DSS needs to be not only accessible to the users (clinicians, auditors, researchers, patients) but it has to rely on standardized concepts expressed in terminologies the users are familar with.
4. **Validation readiness.** In order to guarantee that the provenance metadata being captured is at the right level of granularity and encompasses all the necessary features, the structure of the provenance data needs to be modelled and verified separately from the software implementation.
5. **Traceability of evidence.** The evidence repository will evolve

through the lifecycle of any recommendation software. It is imperative that the content of the repository is subject to an orderly release cycle and an associated quality assurance procedure, including an evidence curation process. This is to ensure that the exact versions of the knowledge bases used in each specific recommendation can be traced back and analysed if needed.

6. **Reproducibility of recommendations.** An underlying feature for a number of these characteristics is that the recommendations made by the system are consistent and reproducible. An identical set of patient cues presented to the same knowledge base and evidence service software have to always yield the same result if there is to be trust in the system. This is the core principle of *reproducibility* which needs to be demonstrable and verifiable.
7. **Responsibility.** While the ultimate responsibility for a diagnosis rests with the user who receives the recommendation and decides what to do with it, in the LHS enabled DSS, the responsibility is shared with the authors of the knowledge base, evidence curators, authors of the reasoning algorithms used, and others. Thus, tracing both the actors using the evidence and the ones generating the evidence is required in order for full accountability to be achieved.
8. **Privacy and security.** Traditionally, security logs have been used to keep track of what is going on in the system and investigate any inappropriate actions. The provenance model needs to go beyond that and be able to demonstrate that the patient data is never used contrary to some set of rules. Furthermore, the transformations and anonymi-

sations on patient data need to be captured in order for the trace to be validated against privacy constraints.

9. **Usability and scalability.** An important feature of provenance support in the DSS is *not to do harm*, and does not impede the normal running of the DSS. This requires seamless integration with no noticeable degradation in performance that would adversely affect the clinician in their daily routines. Furthermore, the system must be able to scale up in line with the expected usage volume, so the provenance store needs to be appropriately specified to cope with accumulation of usage data over time.

These nine requirements were used to guide the design of our provenance solution. We shall now introduce the theory behind provenance templates.

3.2. Provenance templates

Data provenance originated in research communities that rely on uniform computational infrastructures, such as life and earth sciences. The resulting techniques [36] are not directly applicable to LHS scenarios and decision support systems, due to heterogeneity of software systems involved and the need to ensure consistency of provenance graphs produced by different systems. To that end, we introduce provenance templates as abstractions that have domain meaning and can easily be mapped to the actions of the client software tools. The formalism described here is based on W3C PROV [3] as the current standard for representing provenance data as graph models. However, the authors can see no barriers to generalising the approach to any

graph-based provenance representation. [2]

Informally, a provenance template is an abstract provenance graph which may be instantiated to generate a concrete provenance graph, possibly connected to some existing graph structure. We refer to that instantiation process together with associated linkage and validation steps as *graph generation*. The template may contain fragments which are to be repeated, for example, a series of editing operations on some data, and it may specify the places where the generated graph will be grafted (attached) onto some existing graph.

A template, T, is a provenance graph with some reserved annotations, as described in 2.2.1, using a new provenance graph template namespace, `pgt`. A **variable** is a placeholder for the node identifier to be provided during generation process, that uses namespace `var`, e.g. `var:x, var:y, var:z` etc. Nodes in templates may have variable identifiers or normal fixed identifiers. The former are referred to as **variable nodes** and the latter **fixed nodes**. **Value variables** are placeholders for values of attribute-value annotation, rather than node identifiers and use the namespace `vvar`, e.g.`vvar:a`, `vvar:b`, `vvar:c`. Variables are used to identify abstract nodes in the template, and value variables to represent abstract properties associated with either nodes or edges. The scope of variables and value variables is the entire template and each distinct variable or value variable must occur only once in a template. $\mathsf{tvars}(T)$ denotes the set of all variables and value variables occurring in a template T.

Fig. 3 shows a simple template T_1, in diagrammatic form. There, `ent1`

[2]Indeed, the original TRANSFoRm implementation was based on Open Provenance Model [37], a precursor to PROV.

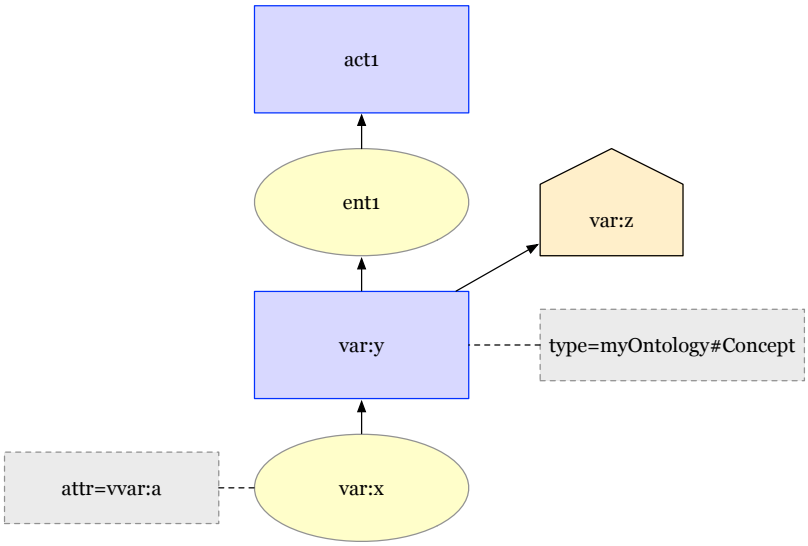

Figure 3: Template T_1

and `act1` are fixed nodes representing respectively a concrete entity and activity, whereas `var:x`, `y` and `var:z` are respectively an entity, an activity and an agent whose identifiers have been replaced by variables. `vvar:a` is a value variable taking the place of a value for the `attr` attribute annotated upon the entity `var:x`.

The node `act1` is annotated with the type value `Concept` taken from ontology `myOntology`. In this way the semantic type of the node is constrained, allowing us to assign clear domain meaning to the concepts in the templates.

3.2.1. Series and parallel zones

An important requirement for our templates is to represent repetition in provenance graphs, often used to describe similar segments that are created by repeated instantiations of a template.

The concept of a repeated pattern in a template is represented using a

zone Z, a connected subgraph that is to be iterated either in series or in parallel upon generation of the graph. The attributes of zones belong to the new `zone` namespace. Each zone has a unique identifier, `zone:id` and may optionally be assigned minimum and maximum bounds `zone:min` and `zone:max`, setting the minimum and maximum number of iterations allowed for the zone. $\mathsf{min}(Z)$ and $\mathsf{max}(Z)$ respectively denote the minimum and maximum bounds of the zone Z, if such values are defined.

A zone is defined by the set of template nodes which belong to it. A node may only belong to one zone. A node that belongs to a zone is denoted an **internal** node N^ι, and its identifier *must* be a variable. Each internal node of a zone is also annotated with the zone identifier using the `pgt:zone` attribute, and inherits the zone's type and bounds. In the figures below, for readability purposes, zones are represented as frames around associated internal nodes, in practice they are still PROV annotations, as described in 2.2.1. A value variable is deemed to belong to a zone if it occurs in an annotation of an internal node of that zone. Let $\mathsf{zvars}(Z)$ denote the set of variables and value variables belonging to a given zone Z.

An **external** node, N^ϵ, is any node of a template that does not belong to a zone. A fixed external node represents a constant node with a fixed value that is not instantiated further. Any external node of a template may also act as a **graft node**, annotated with the `pgt:graft` flag, serving as the point at which the template instance can be linked to another graph. A fixed graft node may share the identifier of a node from a pre-existing graph and similarly a variable graft node may be given an existing node identifier upon substitution. We write $\mathsf{tvars}(T)$ to represent the set of variables and value

variables belonging to external nodes of a template T.

Every edge of a graph has a unique identifier. If the edge is between two internal nodes, it is called an **internal** edge, while an edge between two external nodes of a template T is called an **external** edge. Edges that enter and exit the zone are called **entry** and **exit** edges, respectively. The entry and exit edges of a zone define the manner in which the subgraphs generated by zone iterations are connected to the instantiated external nodes of the template.

A zone may be iterated in parallel or in series, specified by the `zone:type` attribute that can take values of `parallel` or `series` respectively. Intuitively, a parallel iteration represents provenance derivations which may happen independently, where the entry and exit edges of the zone are duplicated to create forking and synchronising points respectively in the final graph, whereas a series type zone represents one which is repeated in sequential fashion and the entry and exit edges define the connection to an initial and terminal state.

A parallel zone must have at least one entry or exit edge in order to ensure graph connectedness upon generation. Series type zones have some additional notation and requirements. A **recursive** edge is a virtual edge of a template by which generated serial iterations of a zone are to be joined. Each such edge defines a connection to be generated from the instantiation of an internal node in one iteration to the instantiation of an internal node in the following iteration. Such an edge is declared by annotating the exit node of the edge with the identifier of the entry node as the value of the `pgt:rec_entry` attribute. The entry node must be another internal node

belonging to the same zone. Write $\mathsf{rec}(Z)$ for the set of recursive edges of a zone Z. Each node given a value for the `pgt:rec_entry` attribute must also be given a value for the `pgt:rec_type` specifying the PROV type of the edge to be created. Each series type zone must have at least one recursive edge to ensure a graph generated from the template is connected.

A template is **valid** if it is a valid provenance graph as defined by [12] and also such that all recursive edges defined in the template also conform to the typing and impossibility constraints applied to normal graph edges.

Fig. 4 shows a larger template T_2, based upon T_1 in which the previous graph has now been identified as a parallel zone, `zone1`. The nodes `ent1` and `act1` are now identified by the variables `var:u` and `var:t` because all internal nodes of a zone must have variables as identifiers. The graph has been extended with new external nodes, some fixed, `ent2`, `act2`, `ent3`, and some variable `var:w` and `var:v`, the last of which has been marked as a graft node. This zone has been annotated with the `zone:min` attribute so as to require a minimum of two iterations upon generation.

Fig. 5 shows another template T_3 in which the original graph has again been identified as a zone but this time of series type. The variable node `var:u` has been annotated with the values for the `pgt:rec_entry` and `pgt:rec_type` attributes, to specify the creation of used edges from each instance of `var:u` to the instance of `var:x` in the following iteration of the zone. The number of iterations of the zone has been limited to eight by use of the `zone:max` attribute. The graph has been expanded with further external nodes in the same way as for Fig. 4.

The resulting provenance graphs generated from these templates are shown

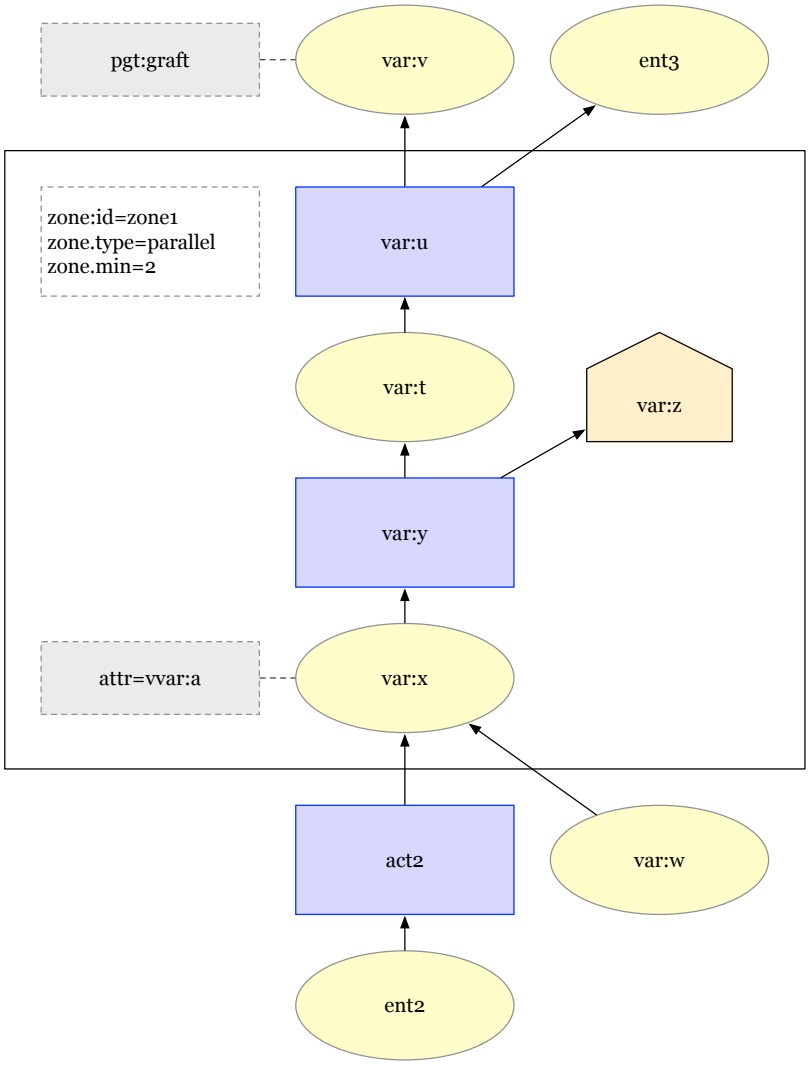

Figure 4: Template T_2 with a parallel zone

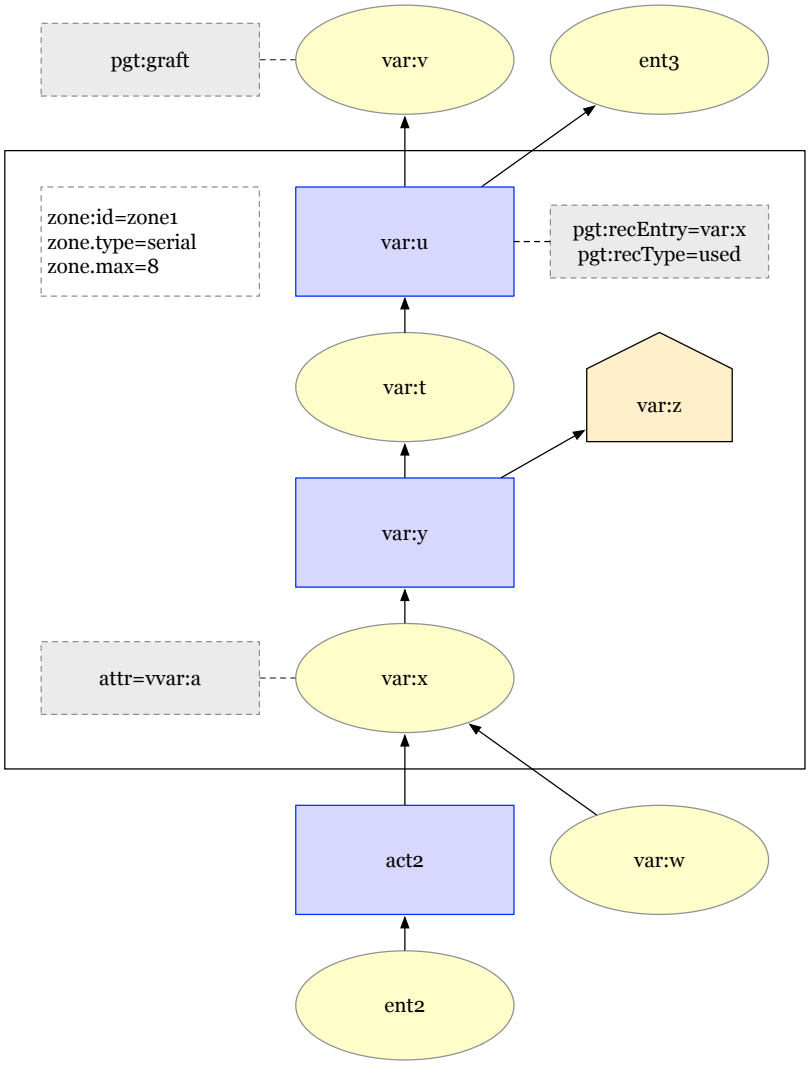

Figure 5: Template T_3 with a series zone

at the end of the next section.

3.3. Template generation

The generation of a particular instantiation of a provenance graph, G, from a template is specified by a **substitution**. A substitution S is defined as a mapping from a pair comprising a qualified name and a non-negative integer representing the **iteration** number to a PROV value. Thus note that no variables or value variables remain after a substitution has been performed. The iteration number for the values substituted for external variables and value variables of a template and for those occurring in the first iteration of any zone is zero. Values substituted for variables or value variables in any subsequent iterations of a zone are numbered sequentially in the obvious manner.

To encode the templates in a standard way, we extend the notation of PROV-N [38] by introducing a new predicate name sub and writing a substitution as a list of expressions of the form $\mathsf{sub}(qn,\ i,\ val)$, where qn is the qualified name of a variable or value variable, i is a non-negative integer and val the value to be substituted for that name in that particular iteration.

In order for a substitution to be **valid**, every variable or value variable has to have at least one value to be substituted and if multiple instantiations of a zone are given, all variables and value variables belonging to that zone must be given a value to be substituted in each iteration, and these iterations must be numbered in increasing order. The total number of iterations to be made for a zone Z specified in a substitution S is written $\mathsf{bound}(Z,\ S)$ and must fall within any given minimum or maximum bound constraints given for the zone, that is, $\mathsf{min}(Z) \leq \mathsf{bound}(Z,\ S) \leq \mathsf{max}(Z)$. Finally, for

each variable, $p \in \mathsf{tvars}(T)$, every value given for p in S *must* be a PROV identifier, which *must* not occur in any pre-existing graph *except* if the node to which v belongs has been labelled as a graft node. (Value variables may be substituted for any PROV value.)

Template generation may proceed in two ways, either in a single-step when given a complete substitution or step-wise using incremental substitutions. Fig. 6 describes the generation of a graph G for a template T given a complete valid substitution S. Graphs are represented as pairs comprising a set of annotated vertices and a set of annotated edges. N_i^ι denotes the copy of the internal node N^ι in the ith iteration of a graph G_i and $\overset{+}{\leftarrow}$ represents the addition of nodes or edges together with any associated annotations to an existing graph as required. The functions `copye` and `copyi` generate a copy of the external nodes, edges and annotations of a template and internal nodes, edges and annotations of a zone respectively.

Generation may also occur in a step-wise fashion, e.g. when a larger template is instantiated through several service calls by the client software. The initial instantiation phase must be executed but then the state of the generated graph may be saved. Instantiations of individual zones may then be executed as needed and the graph state updated at each step. After all zone iterations have been completed a final phase would be executed in which the initial and terminal states of any series zone present are generated and added to the graph. In this scenario, minimum and maximum bounds on zone iterations must be checked after the final phase and the graph state discarded if the conditions are not met. Further implementation details are discussed in Section 4.3.

Data: Template T and Substitution S

Result: Graph G

$T_0 \leftarrow \texttt{copye}(T)$

foreach $p \in \mathit{tvars}(T)$ **do** $p_0 \leftarrow S(p,\ 0)$

$G \overset{+}{\leftarrow} T_0$

foreach $Z \in \mathit{zones}(T)$ **do**

- $k \leftarrow \mathsf{bound}(Z,\ S) - 1$
- **for** $i \leftarrow 0$ **to** k **do**
  - $Z_i \leftarrow \texttt{copyi}(Z)$
  - **foreach** $p \in \mathit{zvars}(Z)$ **do** $p_i \leftarrow S(p,\ i)$
  - $G \overset{+}{\leftarrow} Z_i$
  - **if** $\mathit{type}(Z) = \mathit{parallel}$ **then**
    - **foreach** $(M^{\epsilon},\ N^{\iota}) \in \mathit{entry}(Z)$ **do** $G \overset{+}{\leftarrow} (M,\ N_i)$
    - **foreach** $(M^{\iota},\ N^{\epsilon}) \in \mathit{exit}(Z)$ **do** $G \overset{+}{\leftarrow} (M_i,\ N)$
  - **if** $\mathit{type}(Z) = \mathit{series}$ **then**
    - **if not** $i = 0$ **then**
      - **foreach** $(M,\ N) \in \mathit{rec}(Z)$ **do** $G \overset{+}{\leftarrow} (M_{i-1},\ N_i)$
    - **if** $i = 0$ **then**
      - **foreach** $(M^{\epsilon},\ N^{\iota}) \in \mathit{entry}(Z)$ **do** $G \overset{+}{\leftarrow} (M_0,\ N_1)$
    - **if** $i = k$ **then**
      - **foreach** $(M^{\iota},\ N^{\epsilon}) \in \mathit{exit}(Z)$ **do** $G \overset{+}{\leftarrow} (M_{k-1},\ N_k)$

Figure 6: Generation algorithm

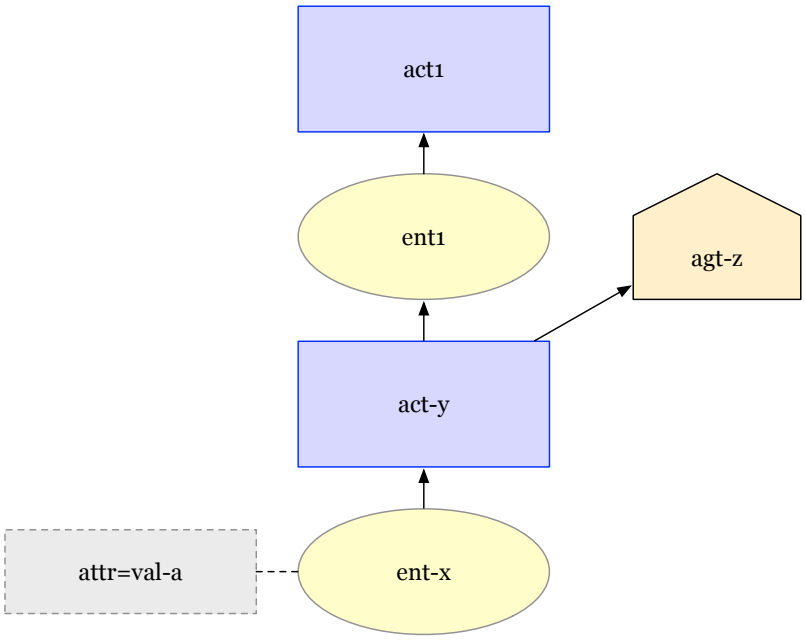

Figure 7: Graph G_1 generated from T_1

3.4. Examples of Generated Graphs

To illustrate the generation process, consider the valid instantiation for template T_1 that is shown in Fig. 7 alongside the corresponding generated graph G_1. As previously noted, the template contains no zones and so all that occurs is the substitution of the external variables and value variables with those identifiers and values given by the instantiation.

Now consider the instantiation and provenance graph shown in Fig. 8 generated from the template T_2 with a parallel type zone given in Fig. 4. Generation of the zone results in the creation of forking nodes at `act1` and `ent-w` and synchronising nodes at `ent-v` and `ent3`. The internal node `var:x` has been instantiated in the two iterations of the zone as `ent-x1` and `ent-x2`

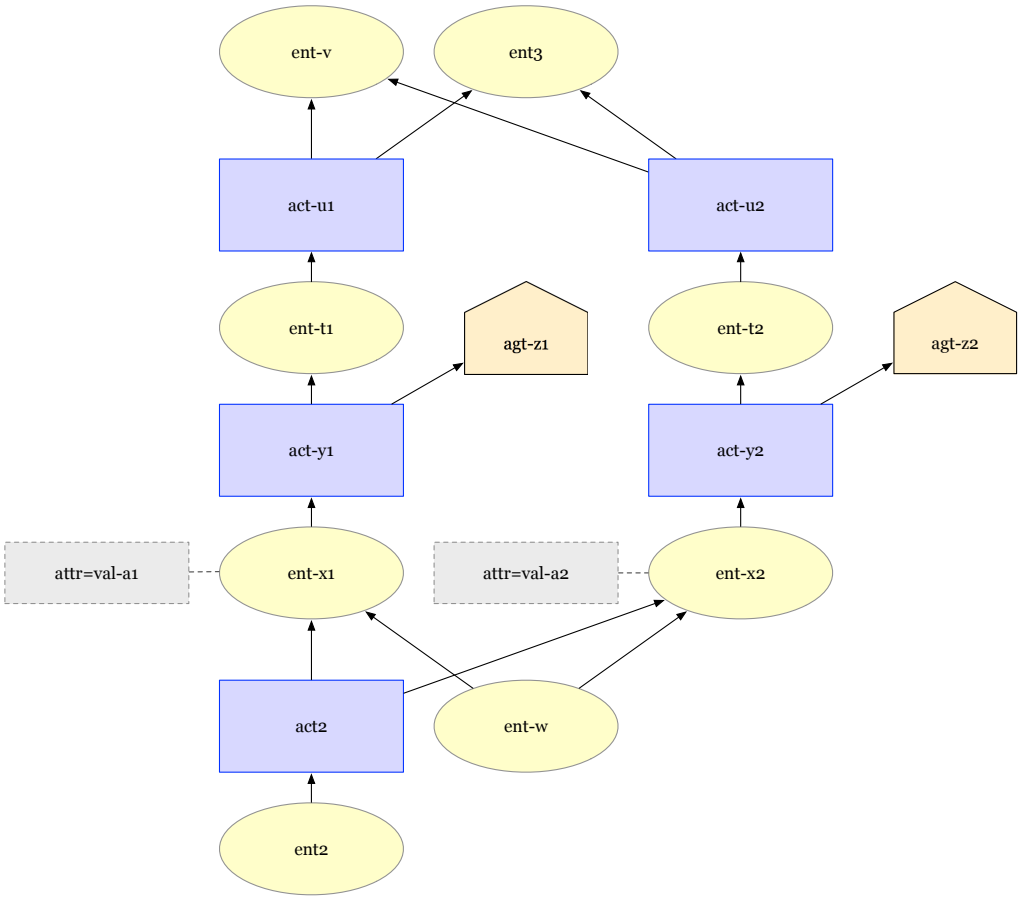

Figure 8: Graph G_2 generated from T_2

and the other variables of the zone are processed similarly. Note that because the node `var:v` was annotated as a graft node, the node `ent-v` may represent a node from a pre-existing provenance graph.

Fig. 9 illustrates the provenance graph generated from the template T_3 given in Fig. 5, using the same instantiation. The series type zone results in the generation of two iterations joined by a recursive edge between the nodes `ent-u1` and `ent-x2`. The nodes `act` and `ent-w` are joined to the first

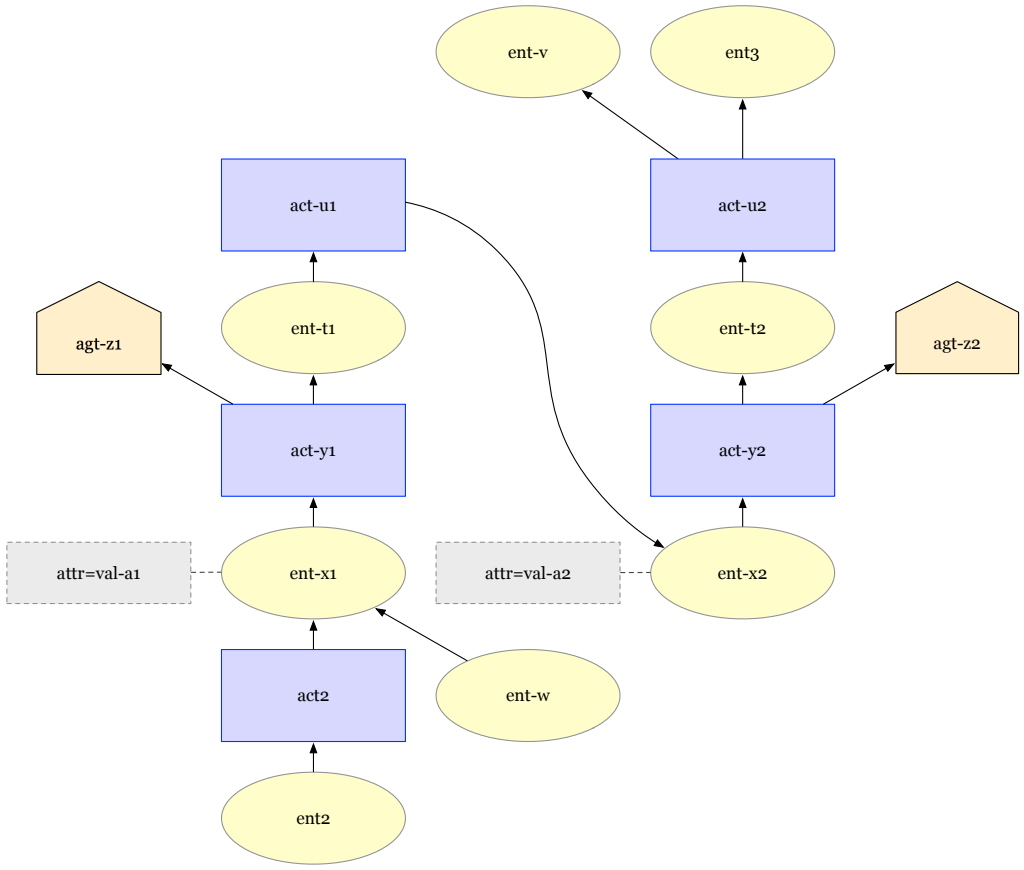

Figure 9: Graph G_3 generated from T_3

iteration and `ent-v` and `ent3` to the final iteration. Again `ent-v` may belong to a pre-existing graph. Instantiation of variables proceeds as for G_2 given in Fig. 8.

4. Results

In order to demonstrate the applicability of the template-based data provenance architecture to providing the relevant audit trail for decision support systems, we have impemented such an architecture within the context

of the TRANSFoRm project. The starting point for defining the provenance use cases was expressing their requirements as a set of basic provenance related questions, describing the provenance information that we require to be automatically recorded and available through our decision support system:

1. **Which decision support user was responsible for initiating a decision support tool session that resulted in a specific diagnostic recommendation being generated on a certain date?** This is a typical audit-style question that assigns responsibility for the diagnosis made.
2. **What authentication was used for a user responsible for a certain action?** This type of question investigates the correctness of the authentication for a particular action.
3. **What clinical evidence cues (presenting symptoms) supported the diagnosis of a particular diagnostic condition?** The input data provided to the diagnostic task needs to be persisted in order to validate the recommendation made.
4. **What clinical evidence data set(s) was a decision support recommendation based on?** In addition to the presenting symptoms, it is also important to understand what evidence base was used to generate the recommendation.
5. **What patients were diagnosed using a particular version of the evidence base?** An example of *taint analysis*, this type of question allows us to find all instances where a potentially incorrect evidence base was used and to trace the affected patients.
6. **Which exact versions of the EHR system and the DSS were**

used in a particular diagnosis? Details about the software tools present in a diagnosis, allowing the user to investigate if there are correlations between certain diagnosis and the software used.

These questions could be asked by an internal or external auditor, either directly, if the role is performed by a system administrator, or via a dedicated user interface. In addition, questions 3. and 4. could be asked by a researcher or clinician wanting to learn more about the guidelines currently in use, via an appropriate user interface. Finally, the latter three questions are highly relevant for investigating potential errors in the system and could be performed by the DSS developer's software team, most likely through a set of direct queries.

One could think of further provenance questions that could be asked about the operation of a decision support system, most notably around the provenance of the evidence base itself and the creation and management of rules therein, however in the TRANSFoRm project, the evidence base was manually curated and thus not suitable for inclusion in our use cases.

4.1. Representing DSS concepts as PROV annotations

One strength of using PROV as the provenance representation language is that it allows for provenance nodes and edges to be annotated with key-value pairs. In order to precisely define the items that are being captured in provenance traces, we have assigned each node an ontological concept and a value, thus allowing provenance graphs to be queried using precise semantics.

The ontological concepts are drawn from three ontologies: TRANSFoRm Software Profile ontology (`TRANSFoRm_SoftwareProfile`) that comprises generic

security and authentication terms, TRANSFoRm Clinical Informatics ontology (`TRANSFoRm_rcto`) that contains clinical research concepts including decision support, and TRANSFoRm Clinical Informatics Provenance ontology (`TRANSFoRm_rctpo`) that maps `TRANSFoRm_rcto` classes onto PROV terms [39]. These ontologies are implemented in OWL and in addition to decision support, cover the full range of Learning Health System concepts in observational studies and clinical trials that were required by TRANSFoRm.

`TRANSFoRm_SoftwareProfile`'s design ensures that each user action in the system can be traced back to the login session during which it happened. This is done using `OpenSession` and `CloseSession` classes and the `SAMLAssertion`, `Session` and `UserName` data entity classes, reflecting the fact that TRANSFoRm used Security Assertion Markup Language (SAML) to implement its security framework. The `OpenSession` and `CloseSession` describe the activities related when a user opens or closes an application. The former activity uses the data provided by authentication services in form of SAML entities, identifying the person accessing the application. This activity generates a `Session` object, which is linked with the following activities during the system execution, including the `CloseSession`.

`TRANSFoRm_rcto` contains classes and relationships relevant to decision support systems, covering clinical evidence and its use in the diagnostic process. These include activities associated with updating and utilization of the clinical evidence repository (`CE_Repository`), and its use by the decision support system (`DSS_system`). While collecting clinical evidence rules, `CollectDiagnosticCues` activities update the `CE_Repository`. During the diagnostic task, a set of diagnostic cues, `CE_PatientDiagnosisCueSet`, are

compiled and used to perform `EvidenceComparison` resulting in a set of matching rules, `CE_MatchingRulesSet` and the final diagnosis recommendation `DSSRecommendation`. `TRANSFoRm_rctpo` creates subclasses of relevant `TRANSFoRm_rcto` classes that are also subclasses of PROV-O ontology [40] concepts, creating identifiers and text labels which are then used as PROV annotations onto provenance template nodes, as shown in Table 1.

4.2. Clinical Decision Support Templates

Two use cases for the TRANSFoRm decision support system were defined and expressed in the form of provenance templates. The first describes the user logging into the system and getting authenticated by the security framework, while the second supports provenance collection during evidence consumption and subsequent clinical recommendation provided by the deployed evidence repository accessed by the decision support tool itself. Note that the two template instantiations are invoked by two different pieces of software in the TRANSFoRm system, the former by the security subsystem and the latter by the decision support tool itself.

In order to represent the semantic categories, each node in the template is further constrained by the ontological annotations described in section 4.1 and shown as PROV key-value attribute pairs in the grey boxes.

The template in Fig. 10 shows the task of a user logging into the decision support system via TRANSFoRm secure middleware, using Security Assertion Markup Language (SAML) authentication and obtaining a session object which is later used to authorise the user to perform actions on the system.

TRANSFoRm concept	Provenance concept
TRANSFoRm_SoftwareProfile#SAMLAssertion	Entity
TRANSFoRm_SoftwareProfile#Session	Entity
TRANSFoRm_SoftwareProfile#User	Agent
TRANSFoRm_SoftwareProfile#UserName	Entity
TRANSFoRm_SoftwareProfile#OpenSession	Activity
TRANSFoRm_SoftwareProfile#CloseSession	Activity
rctpo#CE_Repository	Agent
rctpo#Patient	Agent
rctpo#DSS_system	Agent
rctpo#EHR_system	Agent
rctpo#CollectDiagnosticCues	Activity
rctpo#PatientDiagnosisCueSet	Entity
rctpo#EvidenceComparison	Activity
rctpo#CE_MatchingRulesSet	Entity
rctpo#DSS_Recommendation	Entity

Table 1: TRANSFoRm ontological terms mapped onto provenance concepts in PROV-O

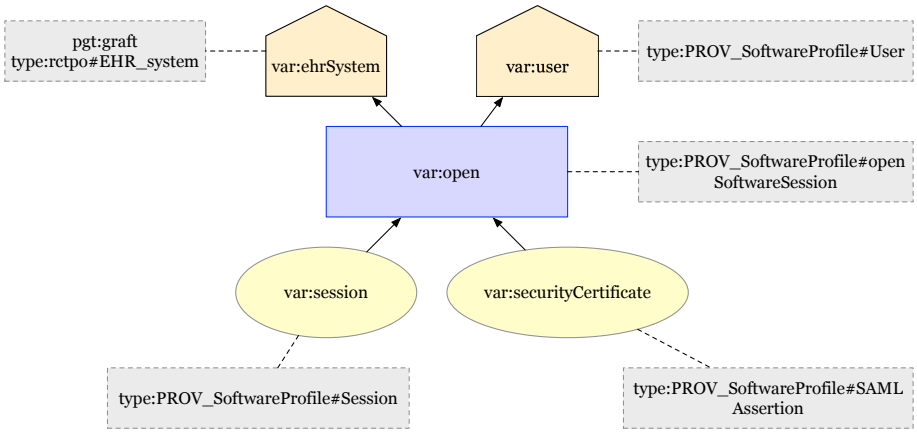

Figure 10: Session template showing the login activity producing a session entity and a security certificate entity

The template in Fig. 11 depicts the operation of the diagnostic decision support system. External nodes `var:ehr` and `var:patient` denote the Electronic Health Record system used and the patient presenting for diagnosis, respectively, while `var:ceRepo` and `var:dss` represent the clinical evidence repository used and the decision support system. The zone represents a single diagnosis task for the patient, of which it is assumed there will be several, with different sets of cues (`var:cueSet`) producing different diagnostic recommendations (`var:diagRec`). Patient symptoms are noted (`var:collectCues`) and used to generate a record of the patient visit `var:patientVisit`, which is used by the decision support system `var:dssSys` to make a comparison (`var:evidenceComp`) against the available clinical evidence in its knowledge base `var:ceRepo` to generate a matching set of rules `var:matchSet` and a diagnostic recommendation `var:diagRec`.

Note that the two templates overlap on the session entity, which is a

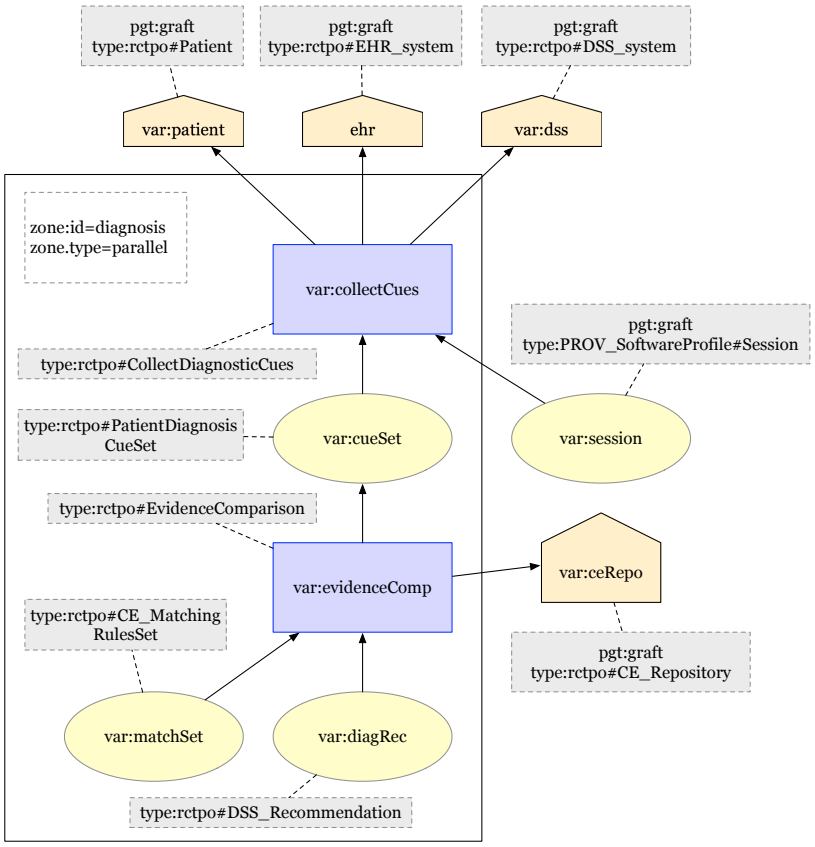

Figure 11: Diagnosis template where a set of diagnosis is made for a patient, each producing diagnostic recommendations

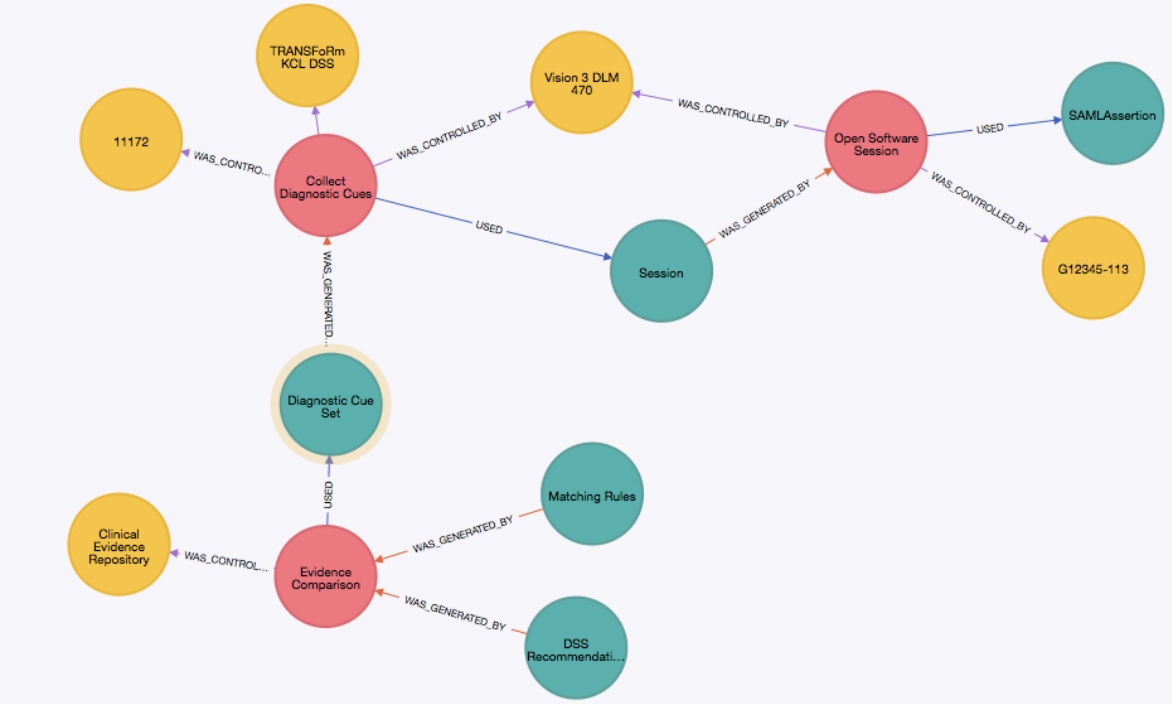

Figure 12: Instantiated template in Neo4J database

graft node in the second template, so there needs to be one login provenance fragment for each diagnosis fragment. One example of the provenance data collected in the TRANSFoRm DSS system is shown in Figure 12, visualised in the Neo4J database.

4.3. Provenance template server architecture

The architecture of the system is illustrated in Figure 13. The overall structure is simple. The provenance server itself is accessible as a web service

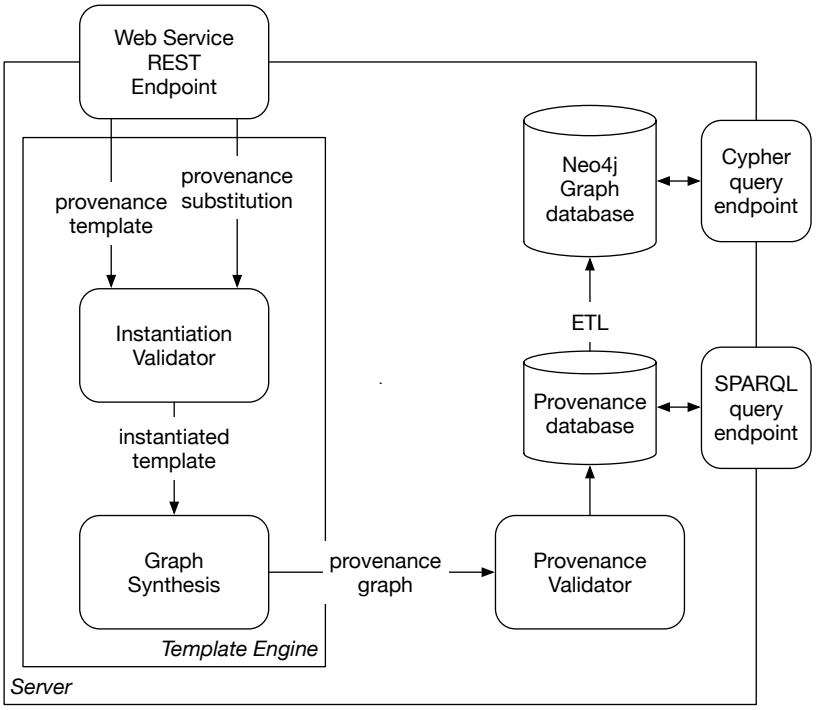

Figure 13: Server Architecture. Graphs stored in the main database are transferred to Neo4J graph database via Extract-Transform-Load (ETL) to facilitate interactive querying.

via an endpoint offering a RESTful API with the data stored using a MySQL relational database with a D2RQ relational-to-RDF adapter [41] allowing querying via SPARQL language that is targetted at semantically annotated data. The data is also transferred into a Neo4j graph database using Extract-Transform-Load (ETL), with an instance of the Neo4j web server allowing captured data to be queried and visualised by users via a browser-based interface using the Cypher query language [42].

The main subcomponent of the server is the template engine, which is

responsible for validating substitutions for a given template and generating the subsequent graph. Provenance graphs are converted to and from the database format by a translator component. Since Neo4j employs a higher-level, graph-theoretic model by a translator component, the ETL represents PROV node types as Neo4j node labels and PROV annotations as Neo4j properties.

As described in section 3.2, the current prototype implementation employs a graph model similar to that of Neo4j. Graphs are described in terms of vertices and edges which may be annotated by dictionaries of named data values. Templating, provenance and user-defined data are kept in separate namespaces. This design allows the engine to be agnostic as regards the provenance standard used by the server, whether that be PROV, the Open Provenance Model (OPM) or any other. However, the remaining components, the REST endpoint, provenance validator and graph translator are specific to a particular standard and must be implemented on a case-by-case basis along with adapters for the template engine.

The main use case is that for storing a graph generated from a template and accompanying instantiation and proceeds as follows. Given a description of the provenance template and instantiation for that template, both are serialised in JSON format and sent to the REST endpoint in a single API call to generate and store the data. After being received by the server, this description is deserialised and then passed to the template engine where it is first validated in order to ensure that the substitution is valid for the template provided. If this succeeds then the graph synthesis component proceeds to generate the expanded graph following the algorithm described in Fig. 6.

This graph is then run through the provenance validator component which checks the validity of the generated graph. If the graph is valid then it is passed to the translator which commits this to the database. Note that if a template includes graft nodes then the provenance validator may need to query the database about existing graphs in order to assess the validity of the generated graph. The second use case, the storage of a complete graph is a subcase of the first. A description of the graph is presented to the server directly via the endpoint at which point it is immediately handed to the provenance validator component rather than to the template engine, and then stored if deemed valid.

As discussed in Section 3.3, it may be also be desirable to generate graphs from a template in a step-wise rather than single-step fashion. In this situation, three API methods would be published. An **initialisation** method would first provide a template and a substitution for external variables and value variables of the template. Following this, one or more calls to a **zone iteration** method would then be made for each zone within the template providing substitution data for the instantiation of a new iteration of the zone. A **finalization** method would then signal that a template was considered complete which would trigger the completion of series type zones, validation of the instantiation and of the generated graph. Intermediate graph states would be commited as temporary graph fragments in the database and either commited fully or rolled back following final validation.

4.4. Provenance data collected

The TRANSFoRm diagnostic decision support system was evaluated using a high-fidelity simulation of the clinical consultation using real EHR

system in a simulated clinic environment. The evaluation employed 34 real physicians and a series of actors reenacting real patient scenarios, around three presenting problems (chest pain, shortness of breath and dyspnea) resulting in a total of 408 patient encounters. Each clinician would be logging into a system once per day, and producing one diagnostic template instantiation with 3-4 zone repetitions per encounter. In a real-world clinic with 8 general practitioners, seeing 40 patients per day, which is standard for an inner London practice, this would translate into around 1000 zone instantiations per day, giving us a scale of the data size and velocity involved in a real life environment.

4.5. Queries developed

The traditional way of querying provenance data uses SPARQL as a semantically enabled query language. In our architecture, the provenance data store accepts SPARQL queries and returns answers. However, we are also interested in the interactive query capabilities, for which the Neo4J database front-end has suitable tooling via its Cypher query language [42] and browsing features. In this type of queries, the reply serves as a starting point for interactive graph exploration, which is not feasible in SPARQL. Thus, in order to demonstrate the viability of using provenance graphs as the audit trail of the decision support tools, we have mapped our initial set of questions onto queries in both SPARQL and the Cypher graph query language, making use of the ontologies we developed. Note that for readability in Cypher queries we are using human readable labels derived from ontological categories.

Query 1: Which decision support user was responsible for initiating a deci-

sion support tool session that resulted in a specific diagnostic recommendation being generated on a certain date?

SPARQL:

```
select ?userName
?oss provo:wasControlledBy ?userName
?oss rdf:type TRANSFoRm_SoftwareProfile:OpenSession
?s provo:wasGeneratedBy ?oss
?s rdf:type TRANSFoRm_SoftwareProfile:Session
?cdc provo:used ?s
?cdc rdf:type rctpo:collectDiagnosticCues
?dcs rdf:type rctpo:diagnosticCueSet
?dcs provo:wasGeneratedBy ?cdc
?ec rdf:type rctpo:EvidenceComparison
?ec provo:used ?dcs
dssRecommendation.getProvURI() provo:wasGeneratedBy ?ec
```

Cypher:

```
MATCH (n:ENTITY {Concept:"DSS Recommendation",
Value:"Appendicitis, unqualified",
Timestamp:"2015-02-24T00:00:00"})-[*]->(m:ENTITY {Concept:"UserNameAgent"})
RETURN m.Value;
```

Query 2: What authentication was used for a user responsible for a certain action?

SPARQL:

```
select ?saml
?oss provo:used ?saml
?oss rdf:type TRANSFoRm_SoftwareProfile:OpenSession
?s provo:wasGeneratedBy ?oss
?s rdf:type TRANSFoRm_SoftwareProfile:Session
action.getProvURI() provo:Used ?s
```

Cypher:

```
MATCH (m:ACTIVITY {Concept:"Collect Diagnostic Cues")-->
(n:ENTITY {Concept:"Session"}) --()->(o:ENTITY {Concept:"SAMLAssertion"})
WHERE ID(m)=346204
RETURN o.Value;
```

Query 3: What clinical evidence cues supported the diagnosis of a particular diagnostic condition?

SPARQL:

```
select ?clEvidenceRepository
datasetRecommendation.getProvURI() provo:wasDerivedFrom ?p
?p rdf:type provo:Activity
?p used ? clEvidenceRepository
?clEvidenceRepository rdf:type rctpo:CE_Repository
```

Cypher:

```
MATCH (n:ENTITY {Concept:"DSS Recommendation",
Value:"Acute pyelonephritis"})-->()-->
(m:ENTITY {Concept:"Diagnostic Cue Set"})
RETURN distinct m.Value;
```

Query 4: What clinical evidence data set(s) was a decision support recommendation based on?

SPARQL:

```
select ?clEvidenceRepository
datasetRecommendation.getProvURI() provo:WasGeneratedFrom ?p
?p rdf:type provo:Activity
?p provo:wasControlledBy ?clEvidenceRepository
?clEvidenceRepository rdf:type rctpo:CE_Repository
```

Cypher:

```
MATCH (n:ENTITY {Concept:"DSS Recommendation"})
-[:WAS_CONTROLLED_BY]->
(m:ENTITY {Concept:"Clinical Evidence Repository"})
WHERE ID(n)=346530
RETURN m.Value;
```

Query 5: What patients were diagnosed using a particular version of the evidence base?

SPARQL:

```
select ?p
?p rdf:type rctpo:Patient
?cdc rdf:type rctpo:collectDiagnosticCues
?cdc provo:wasControlledBy ?p
?dcs rdf:type rctpo:diagnosticCueSet
?dcs provo:wasGeneratedBy ?cdc
?ec rdf:type rctpo:EvidenceComparison
?ec provo:used ?dcs
?ec provo:wasControlledBy ?cer
?cer rdf:type rctpo:clinicalEvidenceRepository
?cer rctpo:hasVersion "2.3"
```

Cypher:

```
MATCH (p:AGENT {Concept:"Patient"}) <-[*]-
(n:ENTITY {Concept:"Evidence Comparison"})-[:WAS_CONTROLLED_BY]-->
(m:AGENT {Concept:"Clinical Evidence Repository"})
WHERE m.Version=2.3
RETURN distinct p.Value;
```

Query 6: Which exact versions of the EHR system and the DSS were used in a particular diagnosis?

SPARQL:

```
select ?ehr_version, ?dss_version
?ehr rctpo:hasVersion ?ehr_version
?dss rctpo:hasVersion ?dss_version
?ehr rdf:type rctpo:EHR_system
```

```
?dss rdf:type rctpo:DSS_system
?cdc rdf:type rctpo:collectDiagnosticCues
?cdc provo:wasControlledBy ?ehr
?cdc provo:wasControlledBy ?dss
?dcs rdf:type rctpo:diagnosticCueSet
?dcs provo:wasGeneratedBy ?cdc
?ec rdf:type rctpo:EvidenceComparison
?ec provo:used ?dcs
dssRecommendation.getProvURI() provo:wasGeneratedBy ?ec
```

Cypher:

```
MATCH (n:ENTITY {Concept:"DSS Recommendation"})-[*]->
(m:AGENT {Concept:"DSS system"}),
n-[*]->(o:AGENT {Concept:"EHR system"})
WHERE ID(n)=346530
RETURN m.Value, o.Value;
```

As can be seen from the structure of implemented queries, the SPARQL queries operating on the RDF representation are, in effect, recreating the structure of the instantiated template to ask questions. Cypher queries meanwhile can make use of generic graph connectivity queries through the `-[*]->` construct, which, while computationally expensive, provides for a more expressive query construction. Furthermore, further navigation and querying from the original result is simpler and faster in Neo4J, which supports the exploratory investigation of provenance traces. Broadly speaking, the strength of Cypher is in processing queries once the entry point has been

found [43], while SPARQL running on RDF representations is better at aggregated queries that need to traverse the entirety of the database. However, with improved indexing capabilities in Neo4J v3, this may be subject to change and we are planning to do the full comparison on a larger, simulated, data set as part of future work.

5. Discussion

In section 3, nine requirements were defined for achieving reproducibility in decision support systems, which we now revisit to demonstrate how our solution addresses them:

1. **System transparency.** The provenance trace in any of its forms provides the insight into the workings of the decision support system, with the granularity defined by the ontologies used, allowing varying levels of detail.
2. **Auditability of recommendations.** To ensure recommendations made are auditable, the system must guarantee that the required subset of information is present in the provenance traces. Templates provide exactly this functionality, by specifying the metadata that will be captured and supporting queries such as Query 3 and Query 5 in section 4.5.
3. **Understandability of data.** Use of domain ontologies (e.g. TRANSFoRm's Clinical evidence ontology) in the provenance node annotations allows queries to be posed in terms of standardised terminologies.
4. **Validation readiness.** Provenance templates are independent of any concrete implementation and the architecture presented is model-

driven. Thus, the model can be validated separately from the software tool.

5. **Traceability of evidence.** In the provenance model we have developed, each version of an evidence base is considered a separate state, and thus is modelled as a separate entity, with each recommendation connected to a single evidence base version. This enables answering questions such as the one posed in Query 4 and Query 6 in section 4.5.
6. **Reproducibility of recommendations.** Rule engines used in decision support systems have to be deterministic if they are to pass validation. Provenance data provides the historical data to conduct that validation and search for the presence of instances where the same input may have resulted in different outputs.
7. **Responsibility.** By storing provenance traces in connected graphs containing the details of the security and authentication mechanisms used, the use of authentication templates guarantees that each action can be traced back to the user or software responsible, as shown in Query 1, Query 2, and Query 6 in section 4.5.
8. **Privacy and security.** The provenance data stored contains the unbroken chain of actions that transformed various pieces of data in the system. If this includes confidential information that should not be presented to the system users, there are several techniques for abstracting parts of provenance data according to predefined security policies [44, 45, 46].
9. **Usability and scalability.** The provenance system is hidden away from the decision support system users, so by implementing light-

weight components and asynchronous REST calls, we ensure that its presence does not impede the clinical consultation process by introducing delays. With regards to the scalability, the use of templates allows us to precisely determine what will be the volume of the provenance data collected and the velocity at which it accumulates, allowing the system administrators to implement appropriate storage policies.

By addressing these, we believe that our approach can be used as a basis for a wide variety of decision support applications. It should be noted that the template design process should always be done in collaboration with the domain experts to ensure correctness and applicability, and that we do not expect the two TRANSFoRm templates to be necessarily sufficient for each possible use case, merely to serve as starting points, together with the ontologies developed. Given that the surrounding architecture is domain-agnostic, we can reasonably expect that it can support use cases at varying levels of complexity, e.g. a production quality implementation would require a conformance testing suite and independent validation of templates developed, ensuring, among other things, that domain ontology constraints are not violated.

The SPARQL and Cypher interfaces allow system developers to quickly query the accrued provenance data, but in order to expose this information to a broader range of end-users, such as institutional and external auditors, commissioning groups, and even clinicians, more visual tooling is necessary. TRANSFoRm implemented two prototype front-end tools: a web interface containing several representative queries such as the ones above, hiding away the complexity and allowing the user to enter only the relevant parameters

and obtain back the results in graph form; and a set of interactive reports containing tabular and chart information, implemented in Eclipse Business Intelligence Reporting Toolkit (BIRT). This approach is currently being explored further by the authors in a follow-up industrial collaboration that shall develop end-user query tools.

Historically, a major challenge for adoption of decision support systems has been the lack of transparency in the recommendations and rules that those recommendations have been based on. In order to evaluate whether provenance technologies can make a tangible difference in clinical practice, and determine whether this information can benefit clinicians directly, provenance elements need to be embedded into a DSS user interface and a full usability study, e.g. using Technical Acceptance Model, shall be necessary. Thus, evaluation of provenance technologies should encompass both the computational and other operational cost involved in running the tools, and the effort needed to design the necessary models and queries should also be measured. Another issue of note is the quality of collected provenance data, its completeness and accuracy, which should be improved by the use of provenance templates. In order to evaluate this, a separate method of data collection should be specified for each provenance question, and the two compared against each other. These and other software engineering challenges related to provenance are being addressed through the work on PRIME methodology [47]. The authors are applying and extending this work within the LHS-Stroke secondary stroke prevention project that is part of the CLAHRC

South London programme[3].

While this paper focused on diagnostic decision support systems, because this was the DSS use case in the TRANSFoRm project, identical issues arise in other types of DSS such as patient management tools and higher level reporting tools such as management portals used by comissioners, insurers, and health administrators. Indeed some of these have been found to have more impact on patient care than diagnostic systems [21].

5.1. Recent developments

The practical need for research into the area of computable provenance for diagnostic decision support has been starkly highlighted by recently reported events in UK general practice surrounding incorrect recommendations being made by a decision support system [48]. The QRISK 2 score is a validated, accepted and widely used decision support aid for predicting the cardiovascular risk in patients in the UK, that is integrated with EHR systems via a parameterised programmatic interface allowing triggering of rules from within the EHR. One particular implementation of the QRISK2 score with a widely used general practice EHR system, was found to be overstating the reported cardiovascular risk in some patients resulting in the wrong guidance to the physician to advise patients to take statin medication to lower that risk. This has resulted in a full investigation by the Medical and Health Products Regulatory Agency (MHRA) in the UK. The initial suspicion has focussed on the communication of patient data between the EHR and the QRISK2 calculator, rather than the implementation of the QRISK2

[3]http://www.clahrc-southlondon.nihr.ac.uk/stroke

tool itself. This has resulted in the need for the EHR vendor in question to identify GP practices to notify potentially impacted patients and to re-examine their cardiovascular risk. The provenance work as described in this research can be seen to be highly applicable in such a logistically difficult scenario. This damage limitation is a classic example of provenance taint analysis as described in section 4, and in such scenario our system could be used to identify:

- Potential GP practices where the QRISK2 score has been used to give diagnostic recommendations. (similar to Query 1, looking into practice instead of a user)
- Diagnostic recommendations actually made for identifiable patients from identifiable EHR systems that actually used the QRISK2 tool. (similar to Query 5)
- The actual patient cue sets that were submitted to the QRISK 2 tool interface to make the diagnostic recommendations (Query 3)
- The returned diagnostic recommendation risk score result for each individual patient involved. (similar to Query 5, including risk score in the template model)

Due to this unfortunate incident, we expect the topics of trust and auditability of decision support systems to come even more to the forefront.

5.2. Tracing evidence evolution

In more generic decision support systems that rely on a collection of rules that are combined to produce recommendations, the questions on the lineage

of rules used to produce a recommendation become important: What data sets and algorithms were used in rule production? How was the rule validation performed? We considered these questions during the TRANSFoRm project, and produced example templates that would satisfy these questions, but they were not implemented at this stage due to the data mining framework remaining separate from the rest of the system. In this extension to the model, each addition, change, or deletion of a rule in the evidence base is tracked, whether the rule has been manually modified or a result of an automated data mining algorithm.

5.3. Related work

The prototype of template-based provenance was introduced in its early form in [49, 50] and successfully demonstrated the feasibility of using provenance templates to support a large, heterogeneous software infrastructure, albeit missing the theoretical foundation and full architecture presented here. A separate effort at Southampton [51] is currently looking into lower-level provenance templates that abstract individual PROV variables, rather than larger graph fragments. The instantiation then proceeds by performing cross-products of all variable value spaces, as restricted by constraints. The concept of substructures in provenance graphs has also been researched in the context of SPARQL queries for RDF provenance repositories and generic graph fragment queries that use a graph motif with a set of constraints on that motif [52]. Related efforts have also been made in the area of graph summarisation [53] which use the graph structure as a basis for summarising and compressing relational knowledge by detecting patterns and compressing structural knowledge encoded within relational graphs, including repetitive or sequen-

tial structures. Finally, a body of work exists in abstracting provenance graphs for security purposes. ZOOM system uses the concept of user views to abstract nodes that are not of interest to the consumer [54]. The TACLP [44], ProvAbs [45], and ProPub [46] approaches use security policy definitions to determine which components of the provenance graph to include and which to abstract.

6. Conclusions and future work

A defining characteristic of the Learning Health System is the trust that must be placed in every aspect of the system [5]. The participants in the LHS must be able to gain insight into its workings if they are to put faith in its actions and entrust it with their data. Furthermore, the system must possess introspective qualities in order to be able to learn about itself and continously improve, engendering a *Virtuous Cycle of Health Improvement*. This implies capabilities for data and knowledge sharing between the research and clinical actors, under clear and automatically enforced privacy and security rules. A semantically clear and unambiguous provenance trace provides a mechanism for such sharing.

This paper has looked into the reproducibility challenges facing decision support systems, guided by the LHS paradigm, and proposed a solution based on data provenance technologies and abstract provenance template constructs. The semantic complexity of the medical domain modelled was modeled using ontologies annotated onto provenance graphs, and the software architecture used the templates to facilitate provenance capture from the decision support tool. The work was originally prototyped in the diagnostic

decision support system developed within the TRANSFoRm project where it was used to capture data from over four hundred simulated diagnostic patient encounters and key analytical queries on that data were shown.

Ultimately, this work contributes to the efforts in integrating trust into computerised decision support systems, enabling transparency and auditability by creating a basis for implementing validation mechanisms. The complexity of decision support systems offers numerous opportunities for problems to arise, from quality of data capture and accuracy of EHR interactions via usability issues to algorithmic errors in rule design. Thus, their increased use puts more and more focus on the techniques for ensuring correctness of the tasks involved. Data provenance offers the mechanism to achieve this, and through use of provenance templates, we have shown how such infrastructure can be implemented in the context of decision support systems.

In the era of Big Data, deep learning systems such as IBM Watson, and other technologies that often rely on black-box analytical environments, it is of paramount importance to support transparency in computerised systems which actions may have direct consequences on human lives. A particularly dangerous assumption of some Big Data evangelists is that with a sufficiently large data, correlation can replace causality in our analytical models. While this may be perfectly fine for market analysis questions such as investigating customer churn or supermarket shopping baskets, medical research in particular depends on full understanding of finer points such as bias, data quality, and statistical significance to derive its conclusions. Thankfully, there is an increasing understanding of the fact that we require not just intelligent machines but intelligible machines [55]. Rather than avoiding Big Data

technologies, we need to understand which aspects of it are well-suited to medical research, and then build research software frameworks that support transparency, auditability, replicability and reproducibility [56].

The version of the DSS provenance infrastructure employed in TRANSFoRm is currently being updated for use by further projects, such as the DSS for prevention of secondary stroke in patients in South London. The tool, developed as part of the CLAHRC South London programme [4], has been designed by the team at King's College London with key stakeholders including clinicians, patients, and commissioners, and the provenance module will provide audibility and traceability of decisions made with the tool. With the recent changes in scalability support in Neo4J, we intend to do away with the relational/RDF store and use Neo4J as our main database storage, providing a SPARQL query front end for backwards compatibility. Furthermore, we plan to add more templates that cover non-diagnostic decision support scenarios and implement some more advanced PROV concepts such as hyperedges representing relations between more than two nodes.

7. Acknowledgements

The work presented here has received funding from EPSRC under grant EP/N027426/1, and the European Unions Seventh Framework Programme for research, technological development and demonstration under grant agreement no 247787. Additional support was received by the National Institute for Health Research (NIHR)Collaboration for Leadership in Applied Health

[4] www.clahrc-southlondon.nihr.ac.uk/

Research and Care programme in South London, based at King's College London. The views expressed are those of the authors and not necessarily those of the NHS, the NIHR or the Department of Health.

References

[1] Editorial, . MHRA investigates problems with cardiovascular risk calculator used in GP practices. The Pharmaceutical Journal 2016;doi:\bibinfo{doi}{10.1211/PJ.2016.20201174}. URL http://www.pharmaceutical-journal.com/news-and-analysis/news-in-brief/mhra-investigates-problems-with-cardiovascular-risk-calculator-used-in-gp-pra 20201174.article.

[2] Friedman, C.P., Wong, A.K., Blumenthal, D.. Achieving a Nationwide Learning Health System. Science Translational Medicine 2010;2(57):29. doi:\bibinfo{doi}{10.1126/scitranslmed.3001456}. URL http://stm.sciencemag.org/content/2/57/57cm29.abstract.

[3] Belhajjame, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., et al. PROV-DM: The PROV Data Model. Tech. Rep.; W3C; 2013. URL http://www.w3.org/TR/prov-dm/.

[4] Delaney, B.C., Curcin, V., Andreasson, A., Arvanitis, T.N., Bastiaens, H., Corrigan, D., et al. Translational Medicine and Patient Safety in Europe: TRANSFoRm-Architecture for the Learning Health System in Europe. BioMed Research International 2015;2015:961526. doi:\bibinfo{doi}{10.1155/2015/961526}.

[5] Friedman, C., Rubin, J., Brown, J., Buntin, M., Corn, M., Etheredge, L., et al. Toward a science of learning systems: a research agenda for the high-functioning Learning Health System. Journal of the American Medical Informatics Association : JAMIA 2015;22(1):43–50. doi:\bibinfo{doi}{10.1136/amiajnl-2014-002977}. URL http://www.ncbi.nlm.nih.gov/pubmed/25342177http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4433378.

[6] De Moor, G., Sundgren, M., Kalra, D., Schmidt, A., Dugas, M., Claerhout, B., et al. Using electronic health records for clinical research: The case of the EHR4CR project. Journal of Biomedical Informatics 2014;doi:\bibinfo{doi}{10.1016/j.jbi.2014.10.006}. URL http://linkinghub.elsevier.com/retrieve/pii/S1532046414002263.

[7] Vogel, J., Brown, J.S., Land, T., Platt, R., Klompas, M.. MDPHnet: secure, distributed sharing of electronic health record data for public health surveillance, evaluation, and planning. American journal of public health 2014;104(12):2265–70. doi:\bibinfo{doi}{10.2105/AJPH.2014.302103}. URL http://www.ncbi.nlm.nih.gov/pubmed/25322301.

[8] Fleurence, R.L., Curtis, L.H., Califf, R.M., Platt, R., Selby, J.V., Brown, J.S.. Launching PCORnet, a national patient-centered clinical research network. Journal of the American Medical Informatics Association : JAMIA 2014;21(4):578–82. doi:\bibinfo{doi}{10.1136/amiajnl-2014-002747}. URL http://www.ncbi.nlm.nih.gov/pubmed/24821743http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4078292.

[9] Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M.R., Li, P., et al. Taverna: A tool for building and running workflows of services. Nucleic Acids Research 2006;34:729–732.

[10] Haasenritter, D.. Making Computations and Publications Reproducible with VisTrails. Computing in Science and Engineering 2012;14(4):18–25.

[11] Seltzer, M., Macko, P.. Provenance Map Orbiter: Interactive Exploration of Large Provenance Graphs. In: Proceedings of the 3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP'11). USENIX Association; 2011, p. 20–21. URL http://dash.harvard.edu/handle/1/5168866.

[12] de Nies, T.. Constraints of the PROV Data Model. Tech. Rep.; World Wide Web Consortium; 2013.

[13] de Dombal, F.T., Leaper, D.J., Staniland, J.R., McCann, A.P., Horrocks, J.C.. Computer-aided diagnosis of acute abdominal pain. British medical journal 1972;2(5804):9–13. URL http://www.ncbi.nlm.nih.gov/pubmed/4552594http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1789017.

[14] Semigran, H.L., Linder, J.A., Gidengil, C., Mehrotra, A.. Evaluation of symptom checkers for self diagnosis and triage: audit study. BMJ (Clinical research ed) 2015;351(1):h3480. doi:\bibinfo{doi}{10.1136/bmj.h3480}. URL http://www.ncbi.nlm.nih.gov/pubmed/26157077http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4496786.

[15] Sim, I., Gorman, P., Greenes, R.A., Haynes, R.B., Kaplan, B., Lehmann, H., et al. Clinical decision support systems for the practice of evidence-based medicine. Journal of the American Medical Informatics Association : JAMIA 2001;8(6):527–34. URL http://www.ncbi.nlm.nih.gov/pubmed/11687560http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC130063.

[16] Musen, M.A., Middleton, B., Greenes, R.A.. Clinical Decision-Support Systems. In: Biomedical Informatics. London: Springer London; 2014, p. 643–674. doi:\bibinfo{doi}{10.1007/978-1-4471-4474-8_22}. URL http://link.springer.com/10.1007/978-1-4471-4474-8{_}22.

[17] Garg, A.X., Adhikari, N.K.J., McDonald, H., Rosas-Arellano, M.P., Devereaux, P.J., Beyene, J., et al. Effects of Computerized Clinical Decision Support Systems on Practitioner Performance and Patient Outcomes. JAMA 2005;293(10):1223. doi:\bibinfo{doi}{10.1001/jama.293.10.1223}. URL http://jama.jamanetwork.com/article.aspx?doi=10.1001/jama.293.10.1223.

[18] Shibl, R., Lawley, M., Debuse, J.. Factors influencing decision support system acceptance. Decision Support Systems 2013;54(2):953–961. doi:\bibinfo{doi}{10.1016/j.dss.2012.09.018}. URL http://linkinghub.elsevier.com/retrieve/pii/S0167923612002539.

[19] Swets, J.A., Dawes, R.M., Monahan, J.. Psychological Science Can Improve Diagnostic Decisions. Psychological Science in the Public Interest 2000;1(1):1–26. doi:\bibinfo{doi}{10.1111/1529-1006.001}. URL http://psi.sagepub.com/lookup/doi/10.1111/1529-1006.001.

[20] Ramnarayan, P., Roberts, G.C., Coren, M., Nanduri, V., Tomlinson, A., Taylor, P.M., et al. Assessment of the potential impact of a reminder system on the reduction of diagnostic errors: a quasi-experimental study. BMC medical informatics and decision making 2006;6:22. doi:\bibinfo{doi}{10.1186/1472-6947-6-22}. URL http://www.ncbi.nlm.nih.gov/pubmed/16646956http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1513379.

[21] Kaplan, B.. Evaluating informatics applications–clinical decision support systems literature review. International journal of medical informatics 2001;64(1):15–37. doi:\bibinfo{doi}{10.1016/s1386-5056(01)00183-6}. URL http://www.ncbi.nlm.nih.gov/pubmed/11673100.

[22] Kawamoto, K., Houlihan, C.A., Balas, E.A., Lobach, D.F., McGlynn, E., Asch, S., et al. Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success. BMJ (Clinical research ed) 2005;330(7494):765. doi:\bibinfo{doi}{10.1136/bmj.38398.500764.8F}. URL http://www.ncbi.nlm.nih.gov/pubmed/15767266http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC555881.

[23] El-Kareh, R., Hasan, O., Schiff, G.D.. Use of health information technology to reduce diagnostic errors. BMJ Quality & Safety 2013;22(Suppl 2):ii40–ii51. doi:\bibinfo{doi}{10.1136/bmjqs-2013-001884}. URL http://qualitysafety.bmj.com/lookup/doi/10.1136/bmjqs-2013-001884.

[24] Patel, V.L., Kaufman, D.R., Kannampallil, T.G.. Diagnostic Reason-

ing and Decision Making in the Context of Health Information Technology. Reviews of Human Factors and Ergonomics 2013;8(1):149–190. doi:\bibinfo{doi}{10.1177/1557234X13492978}. URL http://rev.sagepub.com/lookup/doi/10.1177/1557234X13492978.

[25] Sittig, D.F., Wright, A., Osheroff, J.a., Middleton, B., Teich, J.M., Ash, J.S., et al. Grand challenges in clinical decision support. Journal of biomedical informatics 2008;41(2):387–92. doi:\bibinfo{doi}{10.1016/j.jbi.2007.09.003}. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2660274{\&}tool=pmcentrez{\&}rendertype=abstract.

[26] Blumenthal, D., Tavenner, M.. The Meaningful Use Regulation for Electronic Health Records. New England Journal of Medicine 2010;363(6):501–504. doi:\bibinfo{doi}{10.1056/NEJMp1006114}. URL http://www.nejm.org/doi/abs/10.1056/NEJMp1006114.

[27] Jha, A.K.. Meaningful use of electronic health records: the road ahead. JAMA 2010;304(15):1709–10. doi:\bibinfo{doi}{10.1001/jama.2010.1497}. URL http://www.ncbi.nlm.nih.gov/pubmed/20959581.

[28] Wright, A., Sittig, D.F.. A four-phase model of the evolution of clinical decision support architectures. International Journal of Medical Informatics 2008;77(10):641–649. doi:\bibinfo{doi}{10.1016/j.ijmedinf.2008.01.004}. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2627782{\&}tool=pmcentrez{\&}rendertype=abstract.

[29] Samwald, M., Fehre, K., de Bruin, J., Adlassnig, K.P.. The Arden Syntax standard for clinical decision support: experiences and directions. Journal of biomedical informatics 2012;45(4):711–8. doi:\bibinfo{doi}{10.1016/j.jbi.2012.02.001}. URL http://www.sciencedirect.com/science/article/pii/S1532046412000226.

[30] Wang, D., Peleg, M., Tu, S.W., Boxwala, A.A., Ogunyemi, O., Zeng, Q., et al. Design and implementation of the GLIF3 guideline execution engine. Journal of Biomedical Informatics 2004;37(5):305–318. doi:\bibinfo{doi}{10.1016/j.jbi.2004.06.002}.

[31] Sordo, M., Ogunyemi, O., Boxwala, A.A., Greenes, R.A.. GELLO: an object-oriented query and expression language for clinical decision support. AMIA Annual Symposium proceedings / AMIA Symposium AMIA Symposium 2003;2003:1012. URL http://www.ncbi.nlm.nih.gov/pubmed/14728515http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1480304.

[32] National Academies of Sciences Engineering and Medicine, . How Modeling Can Inform Strategies to Improve Population Health. Washington, D.C.: National Academies Press; 2016. ISBN 978-0-309-37848-2. doi:\bibinfo{doi}{10.17226/21807}. URL http://www.nap.edu/catalog/21807.

[33] Corrigan, D.. Clinical Evidence Ontology. 2015. URL https://bitbucket.org/kclbig/teso/.

[34] Corrigan, D.. An ontology driven clinical evidence ser-

vice providing diagnostic decision support in family practice. AMIA Joint Summits on Translational Science proceedings AMIA Summit on Translational Science 2015;2015:440–4. URL http://www.ncbi.nlm.nih.gov/pubmed/26306282http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4525252.

[35] Moxey, A., Robertson, J., Newby, D., Hains, I., Williamson, M., Pearson, S.A.. Computerized clinical decision support for prescribing: provision does not guarantee uptake. Journal of the American Medical Informatics Association : JAMIA 2010;17(1):25–33. doi:\bibinfo{doi}{10.1197/jamia.M3170}. URL http://www.ncbi.nlm.nih.gov/pubmed/20064798http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2995634.

[36] Missier, P., Paton, N.W., Belhajjame, K.. Fine-grained and efficient lineage querying of collection-based workflow provenance. In: Proceedings of the 13th International Conference on Extending Database Technology, EDBT 2010. ISBN 9781605589459; 2010, p. 299–310. doi:\bibinfo{doi}{10.1145/1739041.1739079}. URL http://portal.acm.org/citation.cfm?id=1739079.

[37] Moreau, L., Freire, J., Futrelle, J., McGrath, R., Myers, J., Paulson, P.. The Open Provenance Model: An Overview. In: Freire, J., Koop, D., Moreau, L., editors. Provenance and Annotation of Data and Processes; vol. 5272 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. ISBN 978-3-540-89964-8; 2008, p. 323–326. URL http://dx.doi.org/10.1007/978-3-540-89965-5{_}31.

[38] Moreau, L., Missier, P.. PROV-N: The Provenance Notation. Tech. Rep.; World Wide Web Consortium; 2013.

[39] Danger, R., Curcin, V.. TRANSFoRm Provenance Ontologies. 2016. URL https://bitbucket.org/kclbig/tpo/.

[40] Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., et al. PROV-O: The PROV Ontology; 2013. URL https://www.w3.org/TR/prov-o/.

[41] Cyganiak, R., Bizer, C., Garbers, J., Maresch, O., Becker, C.. The D2RQ Mapping Language. 2012. URL http://d2rq.org/d2rq-language.

[42] Robinson, I., Webber, J., Eifrem, E.. Graph databases. O'Reilly Media; 2013. ISBN 9781449356262.

[43] Holzschuher, F., Peinl, R.. Performance of Graph Query Languages: Comparison of Cypher, Gremlin and Native Access in Neo4J. In: Proceedings of the Joint EDBT/ICDT 2013 Workshops. EDBT '13; New York, NY, USA: ACM. ISBN 978-1-4503-1599-9; 2013, p. 195–204. doi:\bibinfo{doi}{10.1145/2457317.2457351}. URL http://doi.acm.org/10.1145/2457317.2457351.

[44] Danger, R., Curcin, V., Missier, P., Bryans, J.. Access control and view generation for provenance graphs. Future Generation Computer Systems 2015;49(C):8–27. doi:\bibinfo{doi}{10.1016/j.future.2015.01.014}. URL http://dl.acm.org/citation.cfm?id=2778378.2778560.

[45] Missier, P., Bryans, J., Gamble, C., Curcin, V., Danger, R.. ProvAbs: model, policy, and tooling for abstracting PROV graphs. In: Proceedings of International Provenance and Annotations Workshop (IPAW) 2014. Koln, Germany: Springer; 2014,1406.1998; URL http://arxiv.org/abs/1406.1998.

[46] Dey, S., Zinn, D., Ludäscher, B.. ProPub: Towards a Declarative Approach for Publishing Customized, Policy-Aware Provenance. In: Bayard Cushing, J., French, J., Bowers, S., editors. Scientific and Statistical Database Management; vol. 6809 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. ISBN 978-3-642-22350-1; 2011, p. 225–243. URL http://dx.doi.org/10.1007/978-3-642-22351-8{_}13.

[47] Miles, S., Groth, P., Munroe, S., Moreau, L.. PrIMe: A Methodology for Developing Provenance-Aware Applications. ACM Transactions on Software Engineering and Methodology 2011;20(3):1–42. doi: \bibinfo{doi}{http://dx.doi.org/10.1145/2000791.2000792}.

[48] Alex Matthews-King, . Practices blocked from using CV risk tool on IT system following errors. 2016. URL http://www.pulsetoday.co.uk/your-practice/practice-topics/it/practices-blocked-from-using-cv-risk-tool-on-it-system-following-errors/20031832.fullarticle.

[49] Curcin, V., Danger, R., Kuchinke, W., Miles, S., Taweel, A., Ohmann, C.. Provenance Model for Randomized Controlled Trials. In: Liu, Q., Bai, Q., Giugni, S., Williamson, D., Tay-

lor, J., editors. Data Provenance and Data Management in eScience; vol. 426 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg. ISBN 978-3-642-29930-8; 2013, p. 3–33. doi:\bibinfo{doi}{10.1007/978-3-642-29931-5_1}. URL `http://dx.doi.org/10.1007/978-3-642-29931-5{\_}1`.

[50] Curcin, V., Miles, S., Mercaderes, R.D., Chen, Y., Bache, R., Taweel, A.. Implementing interoperable provenance in biomedical research. Future Generation Comp Syst 2014;34:1–16.

[51] Michaelides, D., Huynh, T.D., Moreau, L.. PROV-TEMPLATE: A Template System for PROV Documents. 2014. URL `https://provenance.ecs.soton.ac.uk/prov-template`.

[52] Lim, C., Lu, S., Chebotko, A., Fotouhi, F.. OPQL: A first OPM-level query language for scientific workflow provenance. Proceedings - 2011 IEEE International Conference on Services Computing, SCC 2011 2011;:136–143doi:\bibinfo{doi}{10.1109/SCC.2011.60}.

[53] Friedman, S.E.. Exploiting Graph Structure to Summarize and Compress Relational Knowledge. In: Proceedings of the 28th International Workshop on Qualitative Reasoning. Minneapolis, MN: SIFT; 2015,.

[54] Biton, O., Cohen-Boulakia, S., Davidson, S.B., Hara, C.S.. Querying and Managing Provenance through User Views in Scientific Workflows. In: 2008 IEEE 24th International Conference on Data Engineering. IEEE. ISBN 978-1-4244-1836-7; 2008, p. 1072–1081. doi:\bibinfo{doi}

{10.1109/ICDE.2008.4497516}. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4497516.

[55] Satya Nadella, . The Partnership of the Future. 2016. URL http://www.slate.com/articles/technology/future{_}tense/2016/06/microsoft{_}ceo{_}satya{_}nadella{_}humans{_}and{_}a{_}i{_}can{_}work{_}together{_}to{_}solve{_}society.html.

[56] Curcin, V.. Embedding data provenance into the Learning Health System to facilitate reproducible research. accepted in Learning Health Systems 2017;1(1).

Appendix A. Full PROV-N specification of graphs shown in Figures

Appendix A.1. Figure 3

```
entity(var:x, [attr=vvar:a])
activity(var:y)
entity(ent1)
agent(var:z)
activity(act1)
wasGeneratedBy(var:x, var:y)
used(var:y, ent1)
wasAssociatedWith(var:y, var:z)
wasGeneratedBy(ent1, act1)
```

Appendix A.2. Figure 4

```
entity(var:x, [attr=vvar:a, zone:id=zone1, zone:type=parallel,
        zone:min=2])
activity(var:y, [zone:id=zone1, zone:type=parallel, zone:min=2])
entity(var:t, [zone:id=zone1, zone:type=parallel, zone:min=2])
agent(var:z, [zone:id=zone1, zone:type=parallel, zone:min=2])
activity(var:u, [zone:id=zone1, zone:type=parallel, zone:min=2])
wasGeneratedBy(var:x, var:y)
used(var:y, ent1)
wasAssociatedWith(var:y, var:z)
wasGeneratedBy(ent1, act1)
entity(ent2)
activity(act2)
entity(var:w)
entity(var:v)
entity(ent3)
entity(var:v, [pgt:graft])
wasGeneratedBy(ent2, act2)
used(act2, var:x)
wasDerivedFrom(var:w, var:x)
used(var:u, var:v)
used(var:u, ent3)
```

Appendix A.3. Figure 5

```
entity(var:x, [attr=vvar:a, zone:id=zone1, zone:type=serial,
        zone:max=8])
```

```
activity(var:y, [zone:id=zone1, zone:type=serial, zone:max=8])
entity(var:t, [zone:id=zone1, zone:type=serial, zone:max=8])
agent(var:z, [zone:id=zone1, zone:type=serial, zone:max=8])
activity(var:u, [zone:id=zone1, zone:type=serial, zone:max=8,
        pgt:recEntry=var:x, pgt:recType:used])
wasGeneratedBy(var:x, var:y)
used(var:y, ent1)
wasAssociatedWith(var:y, var:z)
wasGeneratedBy(ent1, act1)
entity(ent2)
activity(act2)
entity(var:w)
entity(var:v)
entity(ent3)
entity(var:v, [pgt:graft])
wasGeneratedBy(ent2, act2)
used(act2, var:x)
wasDerivedFrom(var:w, var:x)
used(var:u, var:v)
used(var:u, ent3)
```

Appendix A.4. Figure 7

```
entity(ent-x, [attr=val-a])
activity(act-y)
entity(ent1)
agent(agt-z)
```

```
activity(act1)
wasGeneratedBy(ent-x, act-y)
used(act-y, ent1)
wasAssociatedWith(act-y, agt-z)
wasGeneratedBy(ent1, act1)
```

Appendix A.5. Figure 8

```
entity(ent-x1, [attr=val-a1])
activity(act-y1)
entity(ent-t1)
agent(agt-z1)
activity(act-u1)
wasGeneratedBy(ent-x1, act-y1)
used(act-y1, ent-t1)
wasAssociatedWith(act-y1, agt-z1)
wasGeneratedBy(ent-t1, act-u1)
entity(ent-x2, [attr=val-a2])
activity(act-y2)
entity(ent-t2)
agent(agt-z2)
activity(act-u2)
wasGeneratedBy(ent-x2, act-y2)
used(act-y2, ent-t2)
wasAssociatedWith(act-y2, agt-z2)
wasGeneratedBy(ent-t2, act-u2)
entity(ent2)
```

```
activity(act2)
entity(ent-w)
entity(ent-v)
entity(ent3)
wasGeneratedBy(ent2, act2)
used(act2, ent-x1)
used(act2, ent-x2)
wasDerivedFrom(ent-w, ent-x1)
wasDerivedFrom(ent-w, ent-x2)
used(act-u1, ent-v)
used(act-u2, ent-v)
used(act-u1, ent3)
used(act-u2, ent3)
```

Appendix A.6. Figure 9

```
entity(ent-x1, [attr=val-a1])
activity(act-y1)
entity(ent-t1)
agent(agt-z1)
activity(act-u1)
wasGeneratedBy(ent-x1, act-y1)
used(act-y1, ent-t1)
wasAssociatedWith(act-y1, agt-z1)
wasGeneratedBy(ent-t1, act-u1)
entity(ent-x2, [attr=val-a2])
activity(act-y2)
```

```
entity(ent-t2)
agent(agt-z2)
activity(act-u2)
wasGeneratedBy(ent-x2, act-y2)
used(act-y2, ent-t2)
wasAssociatedWith(act-y2, agt-z2)
wasGeneratedBy(ent-t2, act-u2)
entity(ent2)
activity(act2)
entity(ent-w)
entity(ent-v)
entity(ent3)
wasGeneratedBy(ent2, act2)
used(act2, ent-x1)
wasDerivedFrom(ent-w, ent-x1)
used(act-u2, ent-v)
used(act-u2, ent3)
used(act-u1, ent-x2)
```

Appendix A.7. Figure 10

```
entity(var:session, [type:PROV_SoftwareProfile#Session])
entity(var:securityCertificate, [type:PROV_SoftwareProfile#SAMLAssertion])
activity(var:open, [type:PROV_SoftwareProfile#Session])
agent(var:ehrSystem, [pgt:graft, type:rctpo#EHR_system])
agent(var:user, [type:PROV_SoftwareProfile#User])
```

Appendix A.8. Figure 10

```
entity(var:matchSet, [zone:id=diagnosis, zone:type=parallel, type:rctpo#CE_Matchin
entity(var:diagRec, [zone:id=diagnosis, zone:type=parallel, type:rctpo#DSS_Recomme
activity(var:evidenceComp, [zone:id=diagnosis, zone:type=parallel, type:rctpo#Evid
entity(var:cueSet, [zone:id=diagnosis, zone:type=parallel, type:rctpo#PatientDiagn
agent(var:ceRepo, [pgt:graft, type:rctpo#CE_Repository])
activity(var:collectCues, [zone:id=diagnosis, zone:type=parallel, type:rctpo#Colle
entity(var:session, [pgt:graft, type:PROV_SoftwareProfile#Session])
agent(var:patient, [pgt:graft, type:rctpo#Patient])
agent(ehr, [pgt:graft, type:rctpo#EHR_system])
agent(var:dss, [pgt:graft, type:rctpo#DSS_system])
```