# Decentralised Clinical Guidelines Modelling with Lightweight Coordination Calculus

Bo Hu*[1], Srinandan Dasmahapatra[1] , Dave Robertson[2] , Paul Lewis[1]

[1]Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, United Kingdom
[2]Informatics, University of Edinburgh, Edinburgh EH8 9LE, United Kingdom

Email: Bo Hu*- bh@ecs.soton.ac.uk; Srinandan Dasmahapatra - sd@ecs.soton.ac.uk; Dave Robertson - dr@inf.ed.ac.uk; Paul Lewis - phl@ecs.soton.ac.uk;

*Corresponding author

## Abstract

**Background:** Clinical protocols and guidelines have been considered as a major means to ensure that cost-effective services are provided at the point of care. Recently, the computerisation of clinical guidelines has attracted extensive research interest. Many languages and frameworks have been developed. Thus far, however, an enactment mechanism to facilitate decentralised guideline execution has been a largely neglected line of research. It is our contention that decentralisation is essential to maintain a high-performance system in pervasive health care scenarios. In this paper, we propose the use of Lightweight Coordination Calculus (LCC) as a feasible solution. LCC is a light-weight and executable process calculus that has been used successfully in multi-agent systems, peer-to-peer (p2p) computer networks, etc. In light of an envisaged pervasive health care scenario, LCC, which represents clinical protocols and guidelines as message-based interaction models, allows information exchange among software agents distributed across different departments and/or hospitals.

**Results:** We outlined the syntax and semantics of LCC; proposed a list of refined criteria against which the appropriateness of candidate clinical guideline modelling languages are evaluated; and presented two LCC interaction models of real life clinical guidelines.

**Conclusions:** We demonstrated that LCC is particularly useful in modelling clinical guidelines. It specifies the exact partition of a workflow of events or tasks that should be observed by multiple "players" as well as the interactions among these "players". LCC presents the strength of both process calculi and Horn clauses pair of

which can provide a close resemblance of logic programming and the flexibility of practical implementation.

---

## Background

The prevalence of the Internet has slowly but steadily changed the health care industry. One of the most far-reaching outcomes is that an emerging paradigm is gradually reshaping the old "patient-seeing-doctor" scenario into one in which health services and information (e.g. clinical advice and warnings, patient status monitoring, etc.) are decentralised. In line with the WHO's view on "increasing the effectiveness of adherence interventions" [1], this envisioned pervasive health care paradigm offers patients more convenient and personalised health services than ever before and assists in patient's adherence to treatment regimens; and in the meantime it relieves clinicians of many tedious routine jobs and significantly reduces administrative cost. At the heart of the "anywhere and anytime" health care are empowering software agents with decision-making autonomy based on distributed available information. Such a scenario has broached new challenges to the modelling of clinical protocols and guidelines (CPGs): instead of the conventional centralised fashion, guidelines might have to be collaboratively fulfilled by software agents, each only seeing a small fragment of the broad picture. Their local knowledge is then jigsawed together to ensure that the correct protocols have been enforced. A good example that indicates the necessity of decentralised CPG awareness is the situation of comorbidity (e.g. heart disease, AIDS, cancer, diabetes, or mental health). With the development of modern transportation, communication, and tele-medicine, an occurrence of comorbidity might result in the patient being examined and treated concurrently by different experts, in different specialist hospitals, and/or in different regions/countries. How to establish a common understanding with respect to CPGs across different institutions, therefore, becomes a major concern to realise this new health care paradigm. Certainly, such a common understanding requires international efforts and covers multiple research disciplines such as medicine, sociology, psychology, to name but a few. A full account of this is beyond the scope of the paper. Hereinafter, we inspect this issue from knowledge representation point of view: we assume the existence of widely accepted CPGs and focus on a technology that would systematically bring together local agreements amongst distributed clinical services and professionals.

### Clinical guidelines for Chronic Cough and Breast Cancer Triple Assessment

CPGs are "[...]systematically developed statements to assist practitioners and patient decisions about appropriate health care for specific circumstances[...]" [2]. A CPG captures recommendations and regulations that have to be observed when medical investigations and interventions are to be performed. Normally, such information is presented in free text or in semi-structured form and is sometimes reinforced with diagrams, flowcharts, and tables. CPGs have been developed to cover almost every aspect of clinical practice and adopted by a wide variety of medical professionals to ensure the delivery of services with consistent quality and improved cost effectiveness of the health care industry. With the advance in technology, computerised CPGs have begun to attract research and development efforts and serve as the underlying rationale of decision-support systems that are used at the point of care.

### Motivation

The vast interest in formalising CPGs has resulted in a variety of guideline modelling languages providing computer understandable and executable representations wherein guidelines and protocols can be faithfully captured, interpreted and/or enforced [3, 4]. Research has been carried out also on the collaborative aspects of guideline modelling and maintenance, e.g. collaborative editing [5], version management [6], and interoperability [7]. Arguably, the ultimate goal of computerising CPGs is to endow software agents with the capability of practicing or, less ambitiously, of model-checking formal medical procedures on behalf of human experts [8]. Hitherto, this was easily underpinned with centralised guideline models. In the envisaged "next-generation" health care paradigm, however, physically and geographically distributed software agents/human experts might be involved in taking different roles and fulfilling the responsibilities that are allocated to them. Any centralised solutions are evidently inappropriate. Although the existing approaches can accommodate decentralisation to a certain extent, it by no means implies that decentralised guideline enforcement is trivial. A lack of formalisation forces us to distribute tasks among participants by presenting a model to the entire group who then agree upon and mark a partition of the tasks. The *ad hoc* and informal nature of such a partition method provides no guarantee of consistency and repeatability and might raise ethical and quality assurance concerns, thus jeopardising the operational effectiveness of the entire process.

To the best of our knowledge a formalisation of CPG execution in a distributed environment has not been fully explored. The current CPG modelling languages are not designed for distributed tasks and thus do not provide a mechanism to specify unambiguously the task partition/allocation and multiple thread of execution. For instance, in the Breast Cancer Triple Assessment example used by PRO*forma*[1], a single thread workflow is defined. Tasks and actions are specified under top-level *plans*. It, however, does not specify the executor of a plan and how the plan status is preserved, publicised, and shared should it be carried out at different sites by different individuals. It also fails to clearly "tag" the responsibility of individuals (e.g. radiologists) willing to participate in the process, the expected behaviour from these individuals and the communication protocol between the centre on the one side and the individuals on the other side. In other words, CPGs captured in PRO*forma* cannot be used directly to regulate guideline fulfillment in a distributed environment. Our research indicates that a majority of current CPG modelling languages share the same shortcomings (based on the comparative study in [9] and materials of PRODIGY, Asbru, GLARE, GUIDE, and Stepper[2]).

We propose a solution to accommodate the needs of decentralised characteristics in guideline modelling. Instead of taking the conventional "task-network model", we address the inefficiency of decentralisation with respect to current CPG modelling languages by investigating into a formalisation of interactions. One of the exemplary techniques facilitating declarative interaction specification is the Lightweight Coordination Calculus (LCC) [10]. In the rest of this paper, we first set up a list of features that are essential for a CPG modelling language. We then study the applicability of the proposed formalisation by evaluating its capacity against this list of criteria. We further examine LCC by way of examples, i.e. capturing two real-life CPGs as LCC interaction models. Finally, we conclude the paper with possible extensions to LCC that make it more suitable for the problem at hand.

## Methods

Examining whether LCC is suitable for the task of CPG modelling should start with a foundation of essential expressiveness growing out from the desired functionalities of the domain under consideration. In order to compile such a checklist, we adopted the eight axes purposely designed to compare existing CPG modelling frameworks [9]. We further refined/enriched these eight aspects with the general requirements of clinical languages identified by Arnoud and colleagues in their recent study [11]. The final evaluation criteria that are specific to CPG modelling languages are enumerated as follows. Note that these criteria are specific to CPG modelling and should not be generalised to other clinical applications.

- Formalisation and flexibility

[1] http://www.acl.icnet.uk/lab/tallis/Samples.htm

[2] Details of and links to these CPG representation formalisms are available from OpenClinical "Guideline Modelling Methods Summaries" page at http://www.openclinical.org/gmmsummaries.html

A machine-comprehensible guideline modelling language should have formally defined syntax that allows computers to do basic grammar checking and compose well-formed formulae from primitives. The semantics of language constructs should be unambiguously specified, which are interpreted consistently across different models. In the meantime, a candidate modelling language should allow users to define new predicates/functions as a short hand for a set of existing functionalities or for introducing new knowledge.

- Conceptualisation

  Although CPG models should abstract away from implementation idiosyncrasies (e.g. data structure), we would argue that failing to address certain special "needs" in this particular domain might unnecessarily increase the modelling efforts. Such special capabilities include: i) introducing *abstraction and concrete expression* and ii) representing *patient information* and using *medical domain knowledge*. The introduction of abstract medical terms/concepts is essential, so is the manipulation of concrete[3] (e.g. logical, arithmetic and comparison) operators. Meanwhile, a suitable modelling language is expected to have the capability of embedding structured patient data from which decisions and recommendations are drawn.

- Expressiveness and specificity

  The expressiveness of a modelling language refers to what can be left unspoken. The specificity of a language gives the "readiness" of a language in capturing domain-specific (i.e. guideline-specific) knowledge. These two criteria impinge on each other in the the following ways: the specificity might require the language to be equipped with special constructs/operators and primitives that indeed enhance the expressiveness while high expressiveness needs to be tuned to fit in with the CPG domain—the complexity of reasoning with respect to a language might be unnecessarily increased if unwanted expressiveness abounds. We expect a candidate language to be able to express the following components without resorting to external means: 1) *Plans*, including sequential, parallel, cyclical, and iterative plans and *if-then* branching and conditional control; 2) *Goals*, the purpose of the guideline; and 3) *Actions*, the executable components that change the status of an object when the guideline is enforced. On the other hand, we prefer a candidate language to have as simple non-essential functionality as possible to reduce the overall reasoning complexity.

- Readability

[3] The notion of concrete expression is borrowed from Description Logic [12]

A formalised CPG model is meant to be used by not only computers but also human experts. The modelling language, therefore, should be easily comprehended by the users who have only limited or no computer knowledge. Preferably, natural language-like communications are facilitated to reduce the learning curve.

Apart from the above criteria, we also studied LCC with concrete examples. We experimented with the guideline for treating adult chronic cough [13] and the guideline for the Triple Assessment (TA) for Breast Cancer [14] to examine the applicability of LCC in capturing essential CPG information. These CPGs are published by the UK NHS National Library for Health[4]. These two guidelines are selected due to a wide coverage of the workflow control and data manipulation characteristics. The Chronic Cough guideline line is composed mainly in unstructured text while the Triple Assessment guideline complements text with flowcharts and tables. Regarding the target users, the former only concerns general practitioners (family doctors) and nurses while the latter regulates the behaviour of a wide variety of clinical professionals. Although both CPGs are currently used in practice, the former was in effect recently in 2006 while the latter was adopted in 2001 with revisions. We present the resultant LCC guideline models in the next section together with detailed explanations. Note that due to the space limit, only fragments of the LCC guideline models are shown in this paper.

## Results and Discussion

Situated in a decentralised environment, CPG enforcement can be seen as a concurrent process. Among others, process calculi represent the interaction and synchronisation among independent agents through their ability to send and receive messages [15]. Analogously, when proceeding against an established CPG, clinical professionals can rely on message-passing style information exchange (e.g. through patient record, clinical images, etc.) to detect the behaviours of others. This gives us a reasonable inspiration to borrow established formalisms for modelling concurrent systems.

LCC is designed originally for representing coordination between distributed agents. In a multi-agent system, interactions between agents take the form of messages. For instance, an auction is broken down into a series of bidding requests from the bidders and accept/deny responses from the auctioneer. LCC tries to answer the call of formalising such interactions so as to glue software agents together. Although it is not designed for modelling CPGs, LCC clauses can be used to prescribe the behaviours that are allowed in a collaborative environment. For each individual, accepting an LCC interaction model means that

[4] http://www.library.nhs.uk/

he/she is willing to obey message exchanging protocols regulated by the corresponding LCC clauses. That is to say that he/she observes the message passing sequence and accepts the conditions associated with any messages. Meanwhile, messages also serve as triggers to further events. All these fit very well with the envisaged pervasive health care environment. Imagine that a particular patient has to be investigated in separated sites due to unavailability of medical apparatus and/or clinical expertise. Apart from certification/authentication and reputation which only guarantee the general performance of a site rather than its involvement in the current case, one way to ensure other sites act according to protocols is through scheduled communication. Reflecting in computerised modelling languages, scheduled communication is tantamount to sending and expecting messages at pre-defined check-points and restricting the content and format of the messages. The current version of LCC provides suitable tools to tackle messaging-based communications.

Meanwhile, the merit of adopting an existing language/framework is the maturity of mathematical and logic models and the availability of supporting tools. Existing languages bring along with them existing software packages and well-established user communities. In the case of LCC, parsers and visualisation tools have been and are being actively developed, e.g. in the EU-funded OpenKnowledge[5] project for formalising communications in a p2p framework. In this section, we recapitulate the grammar of LCC as well as the semantics of LCC constructs; we explain how LCC meets the requirements set up in the previous section; we present two exemplar LCC applications.

### Syntax and semantics of LCC

Generally speaking, LCC is a process calculus for specifying coordination among multiple participants. It does so by clearly indicating what role an individual plays in a messaging process wherein "roles" are borrowed from institution based systems and reinterpreted as "[...]a form of typing on a process in a process calculus[...]" [10]. An LCC model is built upon the principle that role-playing agents should obey the laws and/or protocols that are explicitly specified against the roles that such agents are expected to take. LCC ensures the fulfillment of roles by individuals through regulating the message-flows among them. These include: the messages that should be sent and are expected to be received and what constraints should be satisfied before a message can be handled. The full picture of LCC syntax is specified in Extended Backus-Naur Form (EBNF) as follows:

[5] http://www.openk.org/

$$
\begin{array}{rcl}
\langle\texttt{Framework}\rangle & := & \{\langle\texttt{Clause}\rangle\ \texttt{,}\}^{1+} \\
\langle\texttt{Clause}\rangle & := & \langle\texttt{Agent}\rangle\ \texttt{::}\ \langle\texttt{Definition}\rangle \\
\langle\texttt{Agent}\rangle & := & \texttt{a(}\ \langle\texttt{Type}\rangle,\ \langle\texttt{ID}\rangle\ \texttt{)} \\
\langle\texttt{Definition}\rangle & := & \langle\texttt{Agent}\rangle \mid \langle\texttt{Message Clause}\rangle \mid \langle\texttt{Definition}\rangle\ \textbf{then}\ \langle\texttt{Definition}\rangle \mid \\
 & & \langle\texttt{Definition}\rangle\ \textbf{or}\ \langle\texttt{Definition}\rangle \mid \langle\texttt{Definition}\rangle\ \textbf{par}\ \langle\texttt{Definition}\rangle \mid \\
 & & null \leftarrow \langle\texttt{Constraint}\rangle \\
\langle\texttt{Message Clause}\rangle & := & \langle\texttt{Message}\rangle \Rightarrow \langle\texttt{Agent}\rangle \mid \langle\texttt{Message}\rangle \Rightarrow \langle\texttt{Agent}\rangle \leftarrow \langle\texttt{Constraint}\rangle \mid \\
 & & \langle\texttt{Message}\rangle \Leftarrow \langle\texttt{Agent}\rangle \mid \langle\texttt{Constraint}\rangle \leftarrow \langle\texttt{Message}\rangle \Leftarrow \langle\texttt{Agent}\rangle \\
\langle\texttt{Constraint}\rangle & := & Term \mid \langle\texttt{Constraint}\rangle \wedge \langle\texttt{Constraint}\rangle \mid \langle\texttt{Constraint}\rangle \vee \langle\texttt{Constraint}\rangle \\
\langle\texttt{Type}\rangle & := & Term \\
\langle\texttt{ID}\rangle & := & Constant \\
\langle\texttt{Message}\rangle & := & Term
\end{array}
$$

In an LCC interaction model, we use predicate $a()$ to specify the role that an individual is playing, $\Rightarrow$ and $\Leftarrow$ to specify the direction of message flow, and $\leftarrow$ for constraints. $Term$ and $Constant$ are implementation-specific. In the current version, $Term$ is a well-formed formula in Prolog logic programming language and $Constant$ is a Prolog constant starting with a lowercase letter.

In order to enable decentralised execution of a CPG strictly based on an LCC interaction model, it is important to provide software agents with a mechanism to unpack LCC clauses, finding the next tasks that it is permitted to perform and updating the status of an interaction accordingly. A set of clause rewriting rules are introduced to ensure LCC constructs are interpreted in a consistent manner [10]. Let $C_i$ be an LCC clause from a model $M$; $I_i$ be a set of received messages currently queueing for an individual participating in an $M$-based interaction; $C_{i+1}$ be the unfolded new LCC clause; $I_{i+1} \subset I_i$ be the set of remaining unprocessed messages; and $O_i$ be the outgoing messages generated when processing $C_i$. An LCC CPG model is interpreted by exhaustively unfolding clauses as detailed in [10] to produce the following sequence:

$$C_1 \xrightarrow{I_1, I_2, M, O_1} C_2, \ldots, C_i \xrightarrow{I_i, I_{i+1}, M, O_i} C_{i+1}, \ldots, C_{n-1} \xrightarrow{I_{n-1}, I_n, M, O_{n-1}} C_n,$$

The interpretation of LCC constraints depends on a particular implementation. In this paper, we assume Prolog as the underlying programming language and thus interpret the constraints in terms of a Prolog logic program. Nevertheless, this by no means deny the possibility of implement LCC constraints with other programming languages, such as JAVA.

$$
\begin{array}{l}
a(\text{on\_call\_doctor}, N) :: \\
\quad routine\_check(P) \Leftarrow a(,A)\ \textbf{then} \\
\quad \left( \begin{array}{l} take\_temperature(P) \Rightarrow a(nurse\,, S)\ \textbf{then} \\ take\_blood\_sample(P) \Rightarrow a(nurse\,, T) \leftarrow \neg blood\_test(P) \end{array} \right)
\end{array}
$$

Pooling together the rewriting rules for LCC-specific constructs and the interpretation of a Prolog program, we obtain the semantics of LCC models. For instance, in the above LCC interaction model, the sequence construct **then** is unfolded by examining the first part of the sequence or, if it is closed (i.e. executed), unfolding the next part. After unfolding, the system tries to instantiate all the variables (e.g. $P$ and $A$) to examine the satisfiability of LCC clauses. A narrative interpretation of this LCC model, therefore, reads "when an on call doctor receives a routine check request on a patient ($P$), he/she first asks an arbitrary nurse ($S$) to take $P$'s body temperature. When the body temperature is done, he/she asks an arbitrary nurse ($T$) to take $P$'s blood sample if $P$ has not been given blood test before."

#### LCC as clinical guideline modelling language

LCC fits very well with the profile set up for a candidate CPG modelling language. In the following, we review LCC-specific characteristics in the context of CPG modelling.

LCC has well-defined syntax and semantics as discussed in the previous section. That is to say that when distributed across different institutes, the perspective users are provided the foundation upon which a unanimous interpretation of LCC clauses can be built as long as the users pledge to endorse an LCC model. Meanwhile, LCC is equipped with the means to express essential CPG components (i.e. *goals*, *plans*, and *actions*). Unfolding an LCC model is a goal-driven process that makes it a perfect candidate preserving the "intention-action" structure of most CPGs. A straightforward approach is to capture the intentions and goals of a CPG as well-formed head/`Agent` of an LCC clause and use the body/`Definition` to detail the expected behaviour and process. An LCC goal might contain more than one sub-goal imitating the nesting of CPG components.

A guideline or protocol might specify sequential, iterative, and concurrent activities. LCC-specific constructs extend Prolog with the capability of naturally representing a wide variety of types of *plans*. Sequential *plans* are guaranteed by the way that an LCC model is rewritten and a Prolog program is interpreted: construct **then** makes sure that `Definition`s are closed in a first-come-first-serve order, while in Prolog, interpretation of sub-goals are decided by the order in which they are introduced. For instance, in Figure 1(a), it is required that the body temperature is taken before blood samples. Combined with constraints, **then** construct allows one to capture switch statements. For instance, line 7 to 15 in Figure 3(b), two alternative routes are regulated by the condition whether a received variable $T$ equals 0. This fragment is equivalent to "*if* $T \neq 0$ *then* ... *else* ...". Parallel *plans* are materialised by LCC construct **par** which connects two or more tasks that are meant to be performed in parallel. In

Figure 1(b), body temperature and blood samples are taken by different clinicians. Note that, performing the two tasks simultaneously is not mandatory as no temporal constraint is given. This, however, by no means implies that temporal constraints cannot be emulated. The current LCC interpreter allows `Constraint`s to be implemented by other programming languages. Temporal and complicated numeric constraints, as well as system calls, can, therefore, be realised through imperative programming languages such as JAVA or purposely designed Prolog libraries. Cyclical *plans* are implemented through recursion. Iterative *plans* can be achieved by cyclical invocations of sub-goals that encapsulate the desired functionalities till the termination or abortion condition is met. In Figure 1(d), blood samples are taken iteratively from a group of patients while list $L$ is used to control maximum number of repeats and termination. In certain circumstances, CPGs give a list of recommendations and allow those who enforce these guidelines to choose from alternatives. Such a style is prevalent when specifying the treatments due to that different medications might have similar or the same effects. For instance when treating acute cough, *menthol* can be "prescribed as menthol crystals BPC or in the form of proprietary capsules." [13] LCC models this with construct **or** as shown in Figure 1(c). When accompanied with switch conditions, **or** is also able to describe mutually exclusive branching of guideline control flow.

Meanwhile, LCC-based interaction adds an extra layer of security to distributed CPG execution. In LCC, coordination is regulated by message passing which suggests that one can leverage the content and format of messages to restrict accesses to confidential and sensible information. Individuals instantiating LCC models are only exposed to the information contained in the messages associated with their roles. This is different from centralised models wherein access control is normally exercised with third-part tools.

Readability of LCC is evident. Predicates can be composed with proper English words and phrased in an informative and self-explained manner. For instance, in Figure 1, short phrases "take temperature" and "take blood sample" are used as task names. Nevertheless, user studies should be carried out to have a better understanding of the readability issue. Human intervention in LCC models is exercised through `Constraint`s: an LCC `Constraint` can be automatically evaluated by software agents or semi-automatically and manually evaluated by human experts through a GUI. Such human driven constraints can be used when confirmation/authorisation is required.

When considering concrete data types, we seek solutions from the underlying programming languages implementing the constraints. Although the capabilities of LCC might be restricted by a particular implementation, some basic Prolog spirits are preserved. These include the convenience of introducing new predicates and the ability to utilise a knowledge base (facts) in representing patient information and domain knowledge. For instance, when recommending an *action*, one can introduce references to medical terminologies (e.g. UMLS[6]) and ontologies (e.g. SNOMED [16]); embed drug interactions in a local knowledge base; and set up a template structure to extract useful information from patient records. The flexibility of satisfying constraints with arbitrary programming languages simplifies the manipulation of arithmetic and basic logical operators, e.g. negation, conjunction, disjunction, etc. All these tools allow one to exploit concrete numbers (e.g. heart rate and body temperature), strings (e.g. patient's full names as the concatenation of her first name and surname and symbolic constants) and comparisons (e.g. $(heart\_rate > 85) \wedge (temp > 40)$). Such knowledge is essential in measuring critical physical conditions of patients and decision criteria in making recommendations (see, for example, Figure 3).

## Case Study

We use two real-life CPGs as examples to explain how LCC interaction models can contribute to the distributed CPG modelling. Some interesting characteristics of these two CPG models are depicted in figures and detailed in this section. We would like to emphasise that the guideline examples demonstrate how one can use LCC to fulfill CPGs with arguments (Figure 2 and 3), collect patient data/information from multiple sources and recommend clinical interventions which are to be carried out at multiple sites (Figure 4), and query patients according to a formal procedure (Figure 5).

### *Breast Cancer Triple Assessment*

In Figure 2, a fragment of the LCC guideline model is illustrated. This LCC model views the Triple Assessment (TA) from the perspective of a radiologist (namely *Rad*). Upon receiving a radiology request from the TA coordinator, *Rad* retrieves routine breast cancer screen results from the TA coordinator or mammography staff if appropriate. Note that certain steps for housekeeping and environment setup are ignored from this LCC model. *Rad* needs to decide how follow-up clinical investigations are performed. According to UK NHS guideline [14], all patients with reported *mass* should undergo further mammography with specific parameters and some patients are given ultrasound should they meet the requirements. This is reflected as two separate task running in parallel whose results are collected as $R2$ and $R3$ respectively. The parameters of mammography are compiled in *Req* and sent to mammography staff alone with patient's identifier. The results from routine screen, ultrasound and well-targeted mammography are aggregated to establish the nature of the breast mass. Subsequently, recommendations

[6] http://umlsinfo.nlm.nih.gov/

which are modelled as a list of **or** alternatives are made based on the radiology investigation.
The example shown in Figure 3 presents a complete interaction model of ultrasound recommendation. During a TA, a proper model of imaging should be recommended to the patient. In some cases, the recommendation of a particular model is drawn from the evaluation of more than one criterion and accumulated as arguments for and against a decision. The ultrasound recommendation model (Figure 3(a)) leverages a cyclic call to repetitively evaluate patient's status and update a numeric score until the termination condition is met. After jumping out of the loop, a final recommendation, based on the accumulative score, is sent back to the radiologist. Note that having single "ultrasound expert" to evaluate all the listed criteria is not mandatory. The ultrasound recommendation model allows such results to be gathered from more than one source as long as every individual taking the role of "ultrasound_expert" accepts and instantiates the LCC data evaluation protocol (as shown in Figure 3(b)). This is facilitated by an anonymous variable denoted using the underscore. In the meantime, an "ultrasound expert" examines and updates received patient records and quantifies his/her decision as an integer. He/she uses the number 0 to signify the termination of the local patient status evaluation procedure.

*Chronic Cough*

Figure 4 illustrates the recommended procedure of an early stage of the engagement with patients having chronic cough. The entry point of Chronic Cough guideline Model is "treating_cough" which is a top-level goal/intention. This fragment of LCC model comprises four components: cough related data gathering, physical examination, three recommended medical investigations which are running in parallel, and HRCT investigation that is performed when other more targeted investigations do not give abnormal findings—each component is enclosed with parentheses. The recommended investigations are running independently in parallel and are performed by different clinical professionals when their respective entry conditions are satisfied. Meanwhile, it is evident that HRCT is only performed when the results of the first three investigations do not reveal the cause of chronic cough. The execution of this model would not reach the HRCT component otherwise.
During the process of diagnosing and treating chronic cough, an individual might frequently refer to others for the information that is unavailable locally. For instance, in Figure 5, a guideline model regulating how and what information should be collected from patients is shown. As recommended in the guideline for adult chronic cough, five different types of information are acquired, including patient's demographic information, smoking habits, characteristic of cough, medications, occupation/hobbies and cough-related

medical history. These information acquisition tasks are considered equally important and are defined/performed in parallel. At the end of each acquisition task, patient's EHR is updated accordingly. When all the parallel-running information acquiring tasks are finished, ehr_collector updates the EHR with patient's confirmation. Cycles present if either an "ehr_collector" feels more information is necessary or details regarding the patients have been changed in the previous acquisition process. When the termination condition is met, "ehr_collector" returns the update EHR as a message sent to $a(\text{treating\_cough}, T)$.

## Conclusions

Medical guidelines have been widely acknowledged as an important means to improve the quality and satisfiability of health care. It provide a tangible and interpretable "template" against which medical practitioners can examine their routines so as to minimise the practice variability and thus reduce the potential cost. The state-of-art languages for CPG modelling perform well in centralised settings. However, a lack of means for task partition and allocation plagues such languages and prevents them from being applied in distributed/pervasive health care scenarios. In this paper, a process calculus based language, LCC, is examined in the context of CPG modelling. We argue that Prolog-based LCC has all the essential functionalities of a guideline modelling language. It further enriches these functionalities with a coordination control mechanism facilitated by message passing. Individuals who pledge to endorse a guideline, therefore, are able to clearly identify their roles, follow the expected behaviours and produce results that are acceptable to others.

The applicability of LCC as a CPG modelling language still requires further investigation in the following directions. Firstly, user studies are necessary to demonstrate the learning curve of LCC. Such users are preferably domain experts who compile guidelines rather than knowledge engineers. Secondly, user friendly parsers and visualisation tools are still under development. Once finished, we can then test guideline models in real hospital settings. Thirdly, LCC guideline models might significantly benefit from common domain knowledge. Capturing such knowledge in a ready-to-use knowledge base and deliver it together with the guideline models is beneficial to both users and developers. Finally, LCC can be enhanced with features such as typed variables, new workflow control constructs (e.g. branching), built-in facilities handling temporal constraints, and assertions with probability. Although it is possible to emulate many of these with the current capacity of LCC (through calls to other programming languages implementing `Constraints`), explicitly introducing them can certainly increase LCC's readability and usability. Other extensions to LCC that are specific to CPG modelling might be identified when more real-life CPGs are rewritten with LCC. This is also the immediate further work that we will commit ourselves to.

## Acknowledgements

This work is supported under the OpenKnowledge and HealthAgents STREP projects funded by EU Framework 6 under Grant numbers IST-FP6-027253 and IST-FP6-027213.

## References

1. World Health Organisation: **Adherence to long-term therapies. Evidence for action** 2003.
2. Field M, Lohr K: *Clinical Practice Guidelines: Directions for a New Program*. Institute of Medicine, Washington D. C. 1990.
3. Musen M, Tu S, Das A, Shahar Y: **EON: a component-based approach to automation of protocol-directed therapy**. *Journal of the American Medical Information Association* 1996, **3**(6):367–388.
4. Sutton D, Fox J: **The Syntax and Semantics of the PROforma Guideline Modeling Language**. *Journal of the American Medical Information Association* 2003, **10**(5):433–476.
5. Shahar Y, Young O, Shalom E, Galperin M, Mayaffit A, Moskovitch R, Hessing A: **A framework for a distributed, hybrid, multiple-ontology clinical-guideline library, and automated guideline-support tools**. *Journal of Biomedical Informatics* 2004, **37**(5):325–344.
6. Terenziani P, Montani S, Bottrighi A, Molino G, Torchio M: **Clinical Guidelines Adaptation: Managing Authoring and Versioning Issues.** In *Proceedings of the 10th Conference on Artificial Intelligence in Medicine in Europe* 2005:151–155.
7. Peleg M, Boxwala A, Bernstam E, Tu S, Greenes R, Shortliffe E: **Sharable representation of clinical guidelines in GLIF: relationship to the Arden syntax**. *Computers and Biomedical Research* 2001, **34**(3):170–181.
8. Groot P, van Harmelen F, Hommersom A, Lucas P, Serban R, ten Teije A: **The Role of Model Checking in Critiquing based on Clinical Guidelines**. In *Proceedings of the Eleventh European Conference on Artificial Intelligence in Medicine (AIME'07)*, Springer Verlag 2007.
9. Peleg M, Tu S, Bury J, Ciccarese P, Fox J, Greenes R, Hall R, Johnson P, Jones N, Kumar A, Miksch S, Quaglini S, Seyfang A, Shortliffe E, Stefanelli M: **Comparing Computer-Interpretable Guideline Models: A Case-Study Approach**. *Journal of the American Medical Information Association* 2003, **10**:52–68.
10. Robertson D: **A Lightweight Coordination Calculus for Agent Systems.** In *Proceedings of the International Workshop on Declarative Agent Languages and Technologies (DALT)* 2004:183–197.
11. van der Maas A, Ter Hofstede H, Ten Hoopen A: **Requirements for Medical Modeling Languages**. *Journal of the American Medical Information Association* 2001, **8**(2):146–162.
12. Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider P (Eds): *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press 2003.
13. Morice A, McGarvey L, Pavord I: *Recommendations for the management of cough in adults*. British Thoracic Society Cough Guideline Group 2006.
14. Wilson R, Asbury D, Cooke J, Michell M, Patnick J: *Clinical Guidelines for Breast Cancer Screening Assessment*. NHS Cancer Screening Programmes 2001.
15. Pierce B: *Foundational Calculi for Programming Languages*, CRC Press 1995 :2190–2207.
16. Cote R, Rothwell D, Ralotay J, Bechett R, Brochu L: **The Systemised Nomenchlature of Medicine: SNOMED International**. *College of American Pathologists, Northfield IL* 1993.

## Figures

$$take\_temperature(P) \Rightarrow a(\_, S) \textbf{ then}$$
$$take\_blood\_sample(P) \Rightarrow a(\_, T)$$

**(a)** ***Sequential execution***

$$take\_temperature(P) \Rightarrow a(\_, S) \textbf{ par}$$
$$take\_blood\_sample(P) \Rightarrow a(\_, T)$$

**(b)** ***Parallel execution***

$$menthol\_crystal\_bpc\,(P) \Rightarrow a(\_, S) \textbf{ or}$$
$$propretary\_capsule\,(P) \Rightarrow a(\_, T)$$

**(c)** ***Nondeterministic choice***

$$\begin{array}{l} a(blood\_sample(L), X) :: \\ \quad (M \Rightarrow a(\_, T) \leftarrow L = [P \mid Rest] \textbf{ then} \\ \quad\; a(blood\_sample(Rest), X)\,) \quad \textbf{or} \\ \quad null \leftarrow L = [\;] \end{array}$$

**(d)** ***Cyclical execution***

Figure 1: LCC constructs

```
a(radiologist, Rad) ::
  radiology_order(P) ⇐ a(ta_coordinator, C) then
  ...
  routine_screen_result(R1) ⇐ a(ta_coordinator, C) or
  routine_screen_result(R1) ⇐ a(mammographer, M) then
  ...
  (  /* evaluate the patient's status to see if a ultrasound is required */
     ( evaluate_ultrasound(P) ⇒ a(ultrasound_recomm(P, R1, 0), Eus) then
       recommend (P, Rec) ⇐ a(ultrasound_recomm(P, _, _), Eus) then
       ( /* If ultrasound is recommended, send off an order */
         null ← equals(Rec, "ultrasound") then
         ultrasound_order(P) ⇒ a(ultrasound_technician, U) ← mass(R1) then
         ultrasound(R2) ⇐ a(ultrasound_technician, U) then )
       or
       null ← equals(Rec, "no ultrasound")   then   null ← R2 = [ ] )

     par

     /* In the meantime, if mass is confirmed,                        */
     /*     order a further mammography with specific requirements */
     ( null ← extract_mass_feature(F, R1) then
       mammo_order_ext(P, Req) ⇒ a(mammographer, M) ← confirm(F) ∧ add(Req, F)∧
                                   add(Req, focal_paddle_compression) ∧ add(Req, cc_view) then
       further_mammography_result(R3) ⇐ a(mammographer, M) )  )   then

  /* evaluate the results so far and save the conclusion in variable A */
  null ← overall_impression (A, R1, R2, R3) then

  (  /* make recommendations accordingly */
     discharge(P) ⇒ a(ta_coordinator, C) ← normal(A)
     or
     discharge(P) ⇒ a(ta_coordinator, C) ← typical_cyst(A) ∧ ¬residual_abnormality(A)
     or
     recommend_aspiration(P) ⇒ a(ta_coordinator, C) ← atypical_cyst(A)
     or
     recommend_needle_biopsy(P) ⇒ a(ta_coordinator, C) ← typical_cyst(A) ∧ residual_abnorm(A)
     or
     ( get_patient_record(P) ⇒ a(ta_coordinator, C) then
       patient_record(H) ⇐ a(ta_coordinator, C) then
       recommend_needle_biopsy(P) ⇒ a(ta_coordinator, C) ← (solid_mass(A) ∨ ¬ exists(H, A))∧
                                                           ¬ adenolipoma(A) ∧ ¬lymphnode(A) )  )
```

Figure 2: Fragment of Triple Assessment Guideline in LCC

$a(\text{ultrasound\_recomm}(P, D, S), M) ::$

$\quad \ldots$

`/* forward patient record to a field expert */`

$\quad \mathit{patient\_ehr}(D) \Rightarrow a(\text{ultrasound\_expert}, \_)$ **then**

`/* accumulate a final score for recommendation */`

$\quad \mathit{score}(T) \Leftarrow a(\text{ultrasound\_expert}, \_)$ **then**

$$\left(\begin{array}{l} \mathit{null} \leftarrow \neg \mathit{equals}(T, 0) \ \textbf{then} \\ \mathit{null} \leftarrow \mathit{update}(S, T) \ \textbf{then} \\ a(\text{ultrasound\_recomm}(P, D, S), M) \end{array}\right)$$

**or**

$$\left(\begin{array}{l} \mathit{null} \leftarrow \mathit{equals}(T, 0) \ \textbf{then} \\ \left(\begin{array}{l} \texttt{/* make final recommendation */} \\ \mathit{recommend}\,(P, \text{``ultrasound''}) \Rightarrow a(\text{radiologist}, \mathit{Rad}) \leftarrow (S \geq 1) \\ \textbf{or} \\ \mathit{recommend}\,(P, \text{``no ultrasound''}) \Rightarrow a(\text{radiologist}, \mathit{Rad}) \leftarrow (S < 1) \end{array}\right) \end{array}\right)$$

**(a)** ***Cyclic LCC model for ultrasound recommendation***

$a(\text{ultrasound\_expert}, X) ::$

$\quad \mathit{patient\_ehr}(D) \Leftarrow a(\text{ultrasound\_recomm}(P, D, S), M)$ **then**

$$\left(\begin{array}{l} \texttt{/* assign scores to different situations */} \\ \mathit{score}(1) \Rightarrow a(\text{ultrasound\_recomm}(P, D, S), M) \leftarrow \mathit{axillary\_lymph\_lump}(D) \ \textbf{then} \ \mathit{update}(D) \\ \textbf{or} \\ \mathit{score}(1) \Rightarrow a(\text{ultrasound\_recomm}(P, D, S), M) \leftarrow \mathit{breast\_implants}(D) \ \textbf{then} \ \mathit{update}(D) \\ \textbf{or} \\ \mathit{score}(1) \Rightarrow a(\text{ultrasound\_recomm}(P, D, S), M) \leftarrow \mathit{localised\_breast\_nodularity}(D) \ \textbf{then} \ \mathit{update}(D) \\ \textbf{or} \\ \mathit{score}(1) \Rightarrow a(\text{ultrasound\_recomm}(P, D, S), M) \leftarrow (\mathit{abnorm}(D) > P3 \wedge \mathit{age}(D) < 35) \ \textbf{then} \ \mathit{update}(D) \\ \textbf{or} \\ \mathit{score}(1) \Rightarrow a(\text{ultrasound\_recomm}(P, D, S), M) \leftarrow \mathit{palpable\_breast\_lump}(D) \ \textbf{then} \ \mathit{update}(D) \\ \textbf{or} \\ \mathit{score}(-99) \Rightarrow a(\text{ultrasound\_recomm}(P, D, S), M) \leftarrow ((\mathit{last\_us}(D) - \mathit{date\_of\_invest}(D)) \leq t) \ \textbf{then} \ \mathit{update}(D) \end{array}\right)$$

**or**

$\quad \mathit{score}(0) \Rightarrow a(\text{ultrasound\_recomm}(P, D, S), M)$

**(b)** ***LCC model for patient data evaluation***

Figure 3: Fragment of LCC Model for recommending ultrasound

$a(\text{treating\_cough}, T) ::$

$\ldots$

`/* retrieve patient record */`

$$\left.\begin{array}{l} \mathit{patient\_data}(\mathit{cough}, P) \Rightarrow a(\text{ehr\_collector}, R) \leftarrow \mathit{find\_patient}(P) \wedge \mathit{find\_data\_provider}(R)\ \mathbf{then} \\ \mathit{cough\_related\_data}(D) \Leftarrow a(\text{ehr\_collector}, R) \end{array}\right. \quad \mathbf{then}$$

`/* order a physical examimantion and get the results */`

$$\left.\begin{array}{l} \mathit{get\_result}(P) \Rightarrow a(\text{physical\_examination}, PE) \leftarrow \mathit{find\_pe}(PE)\ \mathbf{then} \\ \mathit{pe\_result}(PED) \Leftarrow a(\text{physical\_examination}, PE) \end{array}\right. \quad \mathbf{then}$$

$\mathit{null} \leftarrow \mathit{evaluate}(P, \mathit{PED}, \mathit{Result})$ **then**

`/* recommend further tests based on patient's status           */`
`/* there is no specific order in which the tests are performed */`

$$\left(\begin{array}{l} \left(\begin{array}{l} \mathit{chest\_radiography\_order}(P) \Rightarrow a(\text{radiologist}, RA) \leftarrow \mathit{chronic\_cough}(\mathit{Result}) \vee \\ \qquad\qquad\qquad\qquad\qquad\qquad \mathit{atypical\_acute\_cough}(\mathit{Result})\ \mathbf{then} \\ \mathit{radiography\_result}(PRD) \Leftarrow a(\text{radiologist}, RA) \end{array}\right) \\ \\ \mathbf{par} \\ \\ \left(\begin{array}{l} \mathit{spirometry\_order}(P) \Rightarrow a(\text{spirometry\_staff}, SS) \leftarrow \mathit{chronic\_cough}(\mathit{Result})\ \mathbf{then} \\ \mathit{spirometry\_result}(PSD) \Leftarrow a(\text{spirometry\_staff}, SS)\ \mathbf{then} \\ \mathit{prednisolone\_order}(P) \Rightarrow a(\text{trail\_admin}, TA) \leftarrow \mathit{normal}(PSD) \wedge \mathit{asthma\_symptom}(\mathit{Result}) \end{array}\right) \\ \\ \mathbf{par} \\ \\ \left(\begin{array}{l} \mathit{bronchoscopy\_order}(P) \Rightarrow a(\text{bronchoscopy\_staff}, PF) \leftarrow \mathit{foreign\_body\_inhal}(\mathit{Result})\ \mathbf{or} \\ \mathit{bronchoscopy\_order}(P) \Rightarrow a(\text{bronchoscopy\_staff}, BS) \leftarrow \mathit{cause\_unclear}(\mathit{Result})\ \mathbf{then} \\ \mathit{bronchoscopy\_result}(PSD) \Leftarrow a(\text{bronchoscopy\_staff}, BS)\ \mathbf{then} \\ \mathit{prednisolone\_order}(P) \Rightarrow a(\text{trail\_admin}, TA) \leftarrow \mathit{normal}(PSD) \wedge \mathit{asthma\_symptom}(\mathit{Result}) \end{array}\right) \end{array}\right) \quad \mathbf{then}$$

`/* whether HRCT is performed depends on the results of other tests*/`

$$\left(\begin{array}{l} \mathit{hrct\_order}(P) \Rightarrow a(\text{tomography\_staff}, TS) \leftarrow (\mathit{duration\_atypical\_cough}(\mathit{Result}) \geq \mathrm{T}_1) \wedge \\ \qquad\qquad\qquad\qquad\qquad\qquad \mathit{normal}(PRD) \wedge \mathit{normal}(PSD) \wedge \mathit{normal}(PSD)\ \mathbf{then} \\ \mathit{hrct\_result}(PHD) \Leftarrow a(\text{tomography\_staff}, TS) \leftarrow \mathit{hrct\_performed}(P) \end{array}\right) \quad \mathbf{then}$$

$\ldots$

Figure 4: Fragment of Chronic Cough Guideline in LCC

$$
\begin{array}{l}
a(\text{ehr\_collector}(D), R) :: \\
\quad \ldots \\
\quad \mathit{patient\_data}(\mathit{cough}, P) \Leftarrow a(\text{treating\_cough}, T) \ \textbf{then} \\
\\
\quad \texttt{/* acquire cough-related patient data */} \\
\quad \left( \begin{array}{l}
\mathit{get\_sex}(P) \Rightarrow a(\text{patient}, P) \quad \textbf{then} \\
\mathit{update\_ehr}(D, \mathit{Sex}) \leftarrow \mathit{sex}(\mathit{Sex}) \Leftarrow a(\text{patient}, P) \ \textbf{then} \\
\\
\textbf{par} \\
\\
\mathit{get\_age}(P) \Rightarrow a(\text{patient}, P) \quad \textbf{then} \\
\mathit{update\_ehr}(D, \mathit{Age}) \leftarrow \mathit{age}(\mathit{Age}) \Leftarrow a(\text{patient}, P) \\
\\
\textbf{par} \\
\\
\mathit{get\_smoking\_habit}(P) \Rightarrow a(\text{patient}, P) \quad \textbf{then} \\
\mathit{update\_ehr}(D, \mathit{Smoke}) \leftarrow \mathit{smoking\_habit}(\mathit{Smoke}) \Leftarrow a(\text{patient}, P) \\
\\
\textbf{par} \\
\\
\mathit{get\_onset\_cough}(P) \Rightarrow a(\text{patient}, P) \quad \textbf{then} \\
\mathit{update\_ehr}(D, \mathit{Onset}) \leftarrow \mathit{onset}(\mathit{Onset}) \Leftarrow a(\text{patient}, P) \\
\\
\textbf{par} \\
\ldots \\
\textbf{par} \\
\\
\mathit{get\_family\_history}(P) \Rightarrow a(\text{patient}, P) \quad \textbf{then} \\
\mathit{update\_ehr}(D, FH) \leftarrow \mathit{history}(FH) \Leftarrow a(\text{patient}, P)
\end{array} \right) \ \textbf{then} \\
\quad \ldots \\
\quad \texttt{/* repeat if more data is to be acquired */} \\
\quad \left( \begin{array}{l}
\mathit{confirmation}(D) \Rightarrow a(\text{patient}, P) \leftarrow \mathit{changed}(D) \vee \mathit{request\_more\_information}(P) \ \textbf{then} \\
\mathit{confirmed}(D) \Leftarrow a(\text{patient}, P) \ \textbf{then} \\
a(\text{ehr\_collector}(D), R)
\end{array} \right) \\
\\
\quad \textbf{or} \\
\\
\quad \mathit{cough\_related\_data}(D) \Rightarrow a(\text{treating\_cough}, T) \leftarrow \neg\, \mathit{changed}(D)
\end{array}
$$

Figure 5: Fragment of LCC data gathering model for Chronic Cough