Introduction

Knot and link tabulation continues to be a lively area of scientific research and promises to be useful to areas of science such as quantum computing and DNA unpacking [22] [34]. The past 20 years have seen major advances in knot classification, the development of knot invariants, and computational methods for tabulating knots and links. There are tables of the alternating prime knots of up to 23 crossings [28] [48] [49]. While current algorithms provide complete tables of knots (and links), winnowing these tables of duplicates is a time consuming task. Moreover, as knot tables have proved of use to researchers in genetics, and may prove to be to researchers in quantum computation, the need to search these tables in meaningful ways presents itself. For example, of the 4,976,016,485 prime, non-oriented, alternating knots with minimal crossing number of 22, which contain the tangle corresponding to 5/3 (if any)? Of course, knot invariants are useful to distinguish knots. The activity here will develop the newly found strong knot invariant of Meredith and Snyder [32], investigating further characteristics of the knot encoding. In addition, based upon these characteristics, the PI and co-PI will develop a query language for knot theorists capable of efficiently coaxing and prodding mathematically useful information from tables of knots and links. The possibility of using this new knot invariant to make current algorithms more efficient and effective will be investigated as well. In particular, a knot server will be deployed based upon current web standards (XML and XQuery) and spatial logic modelers. One graduate student will research with the team and a Siemens-Westinghouse team of 2-3 highly talented high school students will be mentored. This narrative first gives a brief overview of knot notation systems and of software for knots studies. Then a summary of the π-calculus and spatial logics is provided. The succeeding section then ties the previous two sections together: a description of the Meredith-Snyder knot encoding and its properties. The next section gives the specifications for the promised knot server. The final sections discuss broader impacts, significance, and the educational component.

A brief overview of relevant knot notation systems

The knot notation systems we focus upon herein are the Gauss Code, the Dowker-Thistlethwaite (DT) Code, the Conway Code, and the Master Code of Rankin, Schermann and Smith. Other notation and naming systems exist but don't pertain directly to the proposal.

– Gauss used regular knot projection to arrive at what is now called the Gauss Code of a knot [1]. The Gauss code is the first knot notation system. In 1847,

J.B. Listing classified knots up to 5 crossings by analyzing knot projections. P. G. Tait used an encoding he used in the 1870's to classify knots up to 7 crossings [58]. This encoding is an extension to the Gauss Code. One begins somewhere on the knot projection, proceeds along the knot applying labels to the first, third, fifth etc. crossing until all crossings are labeled; then one traverses the knot once more, writing the label of each crossing in the order that you reach it, attaching a plus or minus sign, depending on whether you are crossing over or under.

- Using Kirkman's classification of certain polygons [31], Tait (and Little, using similar methods) was able to tabulate knots up to 11 crossings [58] [35] [36] [37]. Tait's system included using a simple reduction rewrite strategy within his notation system.
- Reidemeister developed a reduction rewrite system for knot projections (the three Reidemeister moves) [51].
- Conway developed a clever notational system for tangles, finding an algebraic-like system for tangles (the tangle calculus) that led to several methods of reduction rewrites [16]. He used this system to tabulate knots and links of 11 crossings by hand, in one afternoon (an effort that took Tait and Little years of work), discovering one omission and a few duplications in the process. Conway's system can be used to classify all arithmetic (also called algebraic, or rational) knots. The Conway code for a knot originally was given as a basic polyhedron followed by a sorted list of arithmetic tangles. See Conway's paper for details. This system has extensions by Caudron [15] and by Bangert [6].
- Dowker created a variation of Tait's notational system that is easier to implement computationally. Dowker and Thistlethwaite made it the basis for an algorithm that successfully enumerating knots of up to 13 crossings [18]. Not every DT code is valid, i.e. an arbitrary DT code may not correspond to an actual knot, and two distinct composite knots may share the same DT Code. However, a valid DT code for a prime knot specifies the knot uniquely [57]. In their paper, Dowker and Thistlethwaite develop an algorithm to filter out invalid cases. They also give a reduction system to remove duplicates from their enumeration.
- Calvo developed an inductive algorithm, thereby sidestepping the need to check validity of DT codes [13]. However, duplication then becomes a larger issue. Calvo had the insight that understanding the deeper flype structure of prime, non-alternating diagram led to greater efficiencies in his algorithm. The Calvo algorithm was essentially refined in the development of a notational system due to Rankin, Flint, and Schermann , based on what they call the group code which reminds one of the Gauss code, but using a Conway-like insertion scheme to allow for easy reduction of flype structures in the notation [48] [49].

Desirable properties for a knot notation system

Due to complexity considerations, one may well despair of a knot notation system that would allow for classification of all knots. However, in designing a notation system, from the history of knot tabulation and classification we can enumerate the following properties such a notation system may enjoy [1]:

Surjection Each code in the notation represents a knot (or if this is not the case, those codes that do not represent a knot are easily recognizable).

Reduction The notation enjoys a calculus with which to reduce and simplify encodings. Each step of simplification or reduction results in an encoding representing a knot isotopy equivalent to the originating knot.

Minimality The encoding in the notation of a given knot can be reduced to a (finite non-empty set of equivalent) minimal encoding(s).

Injection If a knot has two (or more) minimal reduced encodings, these encodings are transparently equivalent notations for a knot isotopy-equivalent to the original. Important desirable corollary of the previous property: Two knots that have equivalent minimal reduced encodings are isotopy equivalent.

Compositionality Operations on knots correspond, e.g. knot composition, correspond to natural operations on elements in the image of the encoding.

Separation The notation can be used to classify a class X of knots, where X contains a previously classified class of knots (for example, arithmetic knots (also know as algebraic knots) and bracelets) but is not previously classified itself.

Classification The notation enjoys a formal language in which to describe properties and invariants of notation objects that reflect interesting properties of knots. This language should also be useful in selecting specific sets of knots (such as the set of all 21-crossing prime alternating links containing the tangle 3/4). This language ideally should be scaleable and be applicable to tables of indefinite size.

A brief overview of knot software

This is a brief overview of knot software, providing further context as to the relevance of this proposal. A general shortcoming in knot tables is that one cannot search the tables based on a particular criterion, such as: find all the genus 7 knots having minimal crossing number between 15 and 17.

[1] Many of these properties invariably correspond to demands of functoriality on the encoding when considered as a map from the category of knots or braids to some suitable target category while others are demands on the faithfulness (respectively, fullness) of the encoding considered as functor.

The online Table of Knot Invariants [38] lets the user choose one of the knot tables (or subtables) for knots of 12 or fewer crossings and choose from a wide variety of invariants and notations that they wish to see. The resulting table is given with the desired invariants and notations listed for each knot.

The online KnotAtlas [7] contains Mathematica code (the package KnotTheory [8]) and a visual database of knots and links up to 11 crossings.

Rob Scharein's KnotPlot is extensive and extensible software that has a multiplicity of capabilities [56] [57]. It contains a visual database, for example, of all knots up to 10 crossings. KnotPlot also supports tangle calculus and many other features. KnotPlot is open source software and can be used to generate knot representations suitable for import into the PIs' software system. Scharein has built the Knot Server [2], which is intended to data and invariants of knots within the tables of KnotPlot, though at this writing the calculation of invariants is not functional.

N. Imafuji and M. Ochiai [29] developed Knot2000 (K2K), a Mathematica-based package that allows for extensive computation with knots and links. S. Jablan and R. Sazdanovic [52] have used this package and their own webMathematica code to develop the excellent on-line site LinKnot that introduces computational methods in knot tabulation and discusses other aspects of knot theory as well.

The website Knotilus [47] of Rankin et al. has extensive tables of knots and links up to 23 crossings, with browsing of these tables possible via a known Gauss or Dowker-Thistlethwaite code, or via a classification scheme, with a java-based applet for displaying (and drawing) knots.

Gruber [24] has posted online tables of rational knots of up to 16 crossings, though some errors in the tables exist, only a few invariants are calculated, and the tables are not searchable.

Concurrent process calculi and spatial logics

In the last thirty years the process calculi have matured into a remarkably powerful analytic tool for reasoning about concurrent and distributed systems. Process-calculus-based algebraic specification of processes began with Milner's Calculus for Communicating Systems (CCS) [42] and Hoare's Communicating Sequential Processes (CSP) [25] [9] [27] [26], and continue through the development of the so-called mobile process calculi, e.g. Milner, Parrow and Walker's π-calculus [44], Cardelli and Caires's spatial logic [12] [11] [10], or Meredith and Radestock's reflective calculi [40] [41]. The process-calculus-based algebraic specification of processes has expanded its scope of applicability to include the specification, analysis, simulation and execution of processes in domains such as:

– telecommunications, networking, security and application level protocols [3] [4] [30] [33];
– programming language semantics and design [30] [21] [20] [59];
– webservices [30] [33] [39];
– and biological systems [14] [17] [50] [46].

Among the many reasons for the continued success of this approach are two central points. First, the process algebras provide a compositional approach to the specification, analysis and execution of concurrent and distributed systems. Owing to Milner's original insights into computation as interaction [43], the process calculi are so organized that the behavior the semantics of a system may be composed from the behavior of its components [19]. This means that specifications can be constructed in terms of components without a global view of the system and assembled into increasingly complete descriptions. The PIs in this project model knots as being concurrent and distributed systems of crossings. Thus, new computational models of specific classes of knots can be constructed and when they are, thanks to compositionality they can be assembled to extend notations for previous classes (e.g as Conway notation revealed the class of arithmetic knots). In this way, a coherent framework for extending knot notation systems exists. Moreover, there is an underlying mathematical structure in which to search an entirely new population of knot invariants dynamic algebras.

The second central point is that process algebras have a potent proof principle, yielding a wide range of effective and novel proof techniques [45] [53] [54] [55]. In particular, bisimulation encapsulates an effective notion of process equivalence that has been used in applications as far-ranging as algorithmic games semantics [5] and the construction of model-checkers [10]. The essential notion can be stated in an intuitively recursive formulation: a bisimulation between two processes, P and Q, is an equivalence relation, E , relating P and Q, such that whatever action can be observed of P, taking it to a new state P', can be observed of Q, taking it to a new state Q' such that P' is related to Q' by E and vice versa. P and Q, then are bisimilar, if there is some bisimulation relating them. Part of what makes this notion so robust and widely applicable is that it is parameterized in the actions observable of processes, P and Q, thus providing a framework for a broad range of equivalences and up-to techniques all governed by the same core principle [53] [54] [55].

Knots as processes

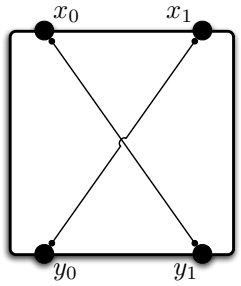
This section contains an overview of Meredith and Snyder's approach [32]. An n crossing knot K is modeled as a system, $[\![K]\!]$, of concurrently executing processes.

More specifically, $[\![K]\!]$ is a parallel composition of $n+1$ processes consisting of n crossing processes and a process constituting a "wiring harness". The latter process can be thought of as the computational equivalent to Conway's "basic polygon," if the knot is in minimal crossing number form. Using the well-established symbolism of mobile process calculi, the process is represented abstractly as follows.

$$[\![K]\!] := (v_0 ... v_{4n-1})(\Pi_{i=0}^{n-1}(\nu\ u)[\![C(i)]\!](v_{4i}, ..., v_{4i+3}, u) \\ |\Pi_{i=0}^{n-1} W(v_{\omega(i,0)}, v_{\omega(i,1)})|W(v_{\omega(i,2)}, v_{\omega(i,3)}))$$

Here, $C(i)$ represents the ith crossing in some regular projection of the knot. The wiring process, $\Pi_{i=0}^{n-1} W(v_{\omega(i,0)}, v_{\omega(i,1)})|W(v_{\omega(i,2)}, v_{\omega(i,3)})$, is itself a parallel composition of wire processes that correspond to edges in the 4-valent graph of the knot shadow. The wiring diagram is constructed from the DT code of the knot projection, as reflected in the indexing function ω. See [32] for the definition of the indexing function). The crossing and wire processes have further substructure, outlined below.

Crossings A crossing is conceived as a process having four possible behaviors, as shown in the defining encoding below and the corresponding diagram.



$$C(x_0, x_1, y_0, y_1, u) := x_1?(s).y_0!(s).(C(x_0, x_1, y_0, y_1, u)|u!) \\ +y_0?(s).x_1!(s).(C(x_0, x_1, y_0, y_1, u)|u!) \\ +x_0?(s).u?.y_1!(s).(C(x_0, x_1, y_0, y_1, u)) \\ +y_1?(s).u?.x_0!(s).(C(x_0, x_1, y_0, y_1, u))$$

A crossing process has four ports x_0, x_1, y_0, y_1 and a hidden synchronizer u. Each port has a partner port, linked as shown in the diagram (note the relationship to Conway's ± 1 tangles [16]). For example, the first behavior (indicated by the first

term of the summand) is that the process listens at port x_1 for a signal s (which will come, if at all, via the wiring process). Having heard s, the signal is passed directly to the port y_0 where the signal is then broadcast via the wiring process. Then the process alerts the hidden synchronizer u that a signal has been passed between the ports, while concurrently preparing itself for further signal processing. The second summand represents a signal passing along the same strand in the opposite direction. The third and fourth summands are similar to the first two, except that before passing any received signal to its partner port, the process waits for a signal from the synchronizer u before allowing the signal to pass. So the role of u is that of a traffic controller who gives priority to traffic over the route between x_1 and y_0, mimicking an over-crossing.

Wirings As an illustration of the expressive power of the formalism, taken together with the short description of the calculus in the next section, the definitions below fully equip the interested reader to verify that wire processes are lossless, infinite capacity buffers.

$$\begin{aligned} W(x,y) &:= (\nu\ n\ m)(Waiting(x,n,m)|Waiting(y,m,n)) \\ Waiting(x,c,n) &:= x?(v).(\nu\ m)(Cell(n,v,m)|Waiting(x,c,m)) \\ &\quad +c?(w).c?(c).Ready(x,c,n,w) \\ Ready(x,c,n,w) &:= x?(v).(\nu\ m)(Cell(n,v,m)|Ready(x,c,m,w)) \\ &\quad +x!(w).Waiting(x,c,n) \end{aligned}$$

The distinguishing power of dynamics In summary, to each knot, K, the encoding $[\![-]\!] : Knots \to \pi\text{-}calculus$, associates an invariant, $[\![K]\!]$, an expression in a calculus of message-passing proceses. Of note, the notion of equivalence of knots, ambient isotopy, denoted $\sim$, coincides perfectly with the notion of equivalence of processes, i.e. bisimulation, denoted $\simeq$. Stated more formally,

$$K_1 \sim K_2 \iff [\![K_1]\!] \simeq [\![K_2]\!]$$

(For the proof, see [32].) In other words, unlike other invariants, the coincidence of process dynamics with knot characteristics enables it to be perfectly distinguishing. As dicussed below, among the other beneficial consequences of this coincidence is the application of process logics, especially the spatial sub-family of the Hennessy-Milner logics, to reason about knot characteristics and knot classes.

The syntax and semantics of the notation system

Having bootstrapped some intuitive account of the target calculus of via the encoding, we summarize its technical presentation below. The popular presentation of these calculi follows a generators and relations style. The grammar, below, describing term constructors, freely generates the set of processes. This set is then quotiented by a relation known as structural congruence.

$$\text{SUMMATION } M, N ::= 0 \mid x.A \mid M + N \qquad A ::= (\boldsymbol{x})P \mid [\boldsymbol{x}]P \text{ AGENT}$$

$$\text{PROCESS } P, Q ::= N \mid P|Q \mid X\langle \boldsymbol{y} \rangle \mid (\mathsf{rec}\ X(\boldsymbol{x}).P)\langle \boldsymbol{y} \rangle \mid (\nu\ \boldsymbol{x})P$$

Note that the presentation uses $\boldsymbol{x}$ to denote *lists* of names of length $|\boldsymbol{x}|$. In the encodings for crossings and wires given above we adopted the following standard abbreviations.

$$x?(\boldsymbol{y}).P \triangleq x.(\boldsymbol{y})P \qquad x!(\boldsymbol{y}).P \triangleq x.[\boldsymbol{y}]P \qquad X(\boldsymbol{y}) := P \triangleq (\boldsymbol{y})(\mathsf{rec}\ X(\boldsymbol{x}).P)\langle \boldsymbol{y} \rangle$$

$$\Pi_{i=0}^{n-1} P_i \triangleq P_0 | \ldots | P_{n-1}$$

Structural congruence

Definition 1. *The* structural congruence*, $\equiv$, between processes is the least congruence closed with respect to alpha-renaming, satisfying the abelian monoid laws (associativity, commutativity and* 0 *as identity) for parallel composition as well as summation, and the following axioms:*

$$(\nu\ x)0 \equiv 0 \qquad (\nu\ x)(\nu\ x)P \equiv (\nu\ x)P \qquad (\nu\ x)(\nu\ y)P \equiv (\nu\ y)(\nu\ x)P$$

$$P|(\nu\ x)Q \equiv (\nu\ x)(P|Q),\ \textit{if}\ x \notin \mathcal{FN}(P)$$

$$(\mathit{rec}\ X(\boldsymbol{x}).P)\langle \boldsymbol{y} \rangle \equiv P\{\boldsymbol{y}/\boldsymbol{x}\}\{(\mathit{rec}\ X(\boldsymbol{x}).P)/X\}$$

Operational semantics

Finally, we introduce of the computational dynamics. What marks these algebras as distinct from other more traditionally studied algebraic structures, e.g. vector spaces or polynomial rings, is the manner in which dynamics is captured. In traditional structures dynamics is expressed through morphisms between such structures, as in

linear maps between vector spaces or morphisms between rings. In algebras associated with the semantics of computation, the dynamics is expressed as part of the algebraic structure, itself, through a reduction reduction relation, typically denoted by $\rightarrow$. Below we find a recursive presentation of this relation for the calculus used in the encoding.

$$\textsc{Comm}\ \frac{\boldsymbol{y} \cap \boldsymbol{v} = \emptyset \qquad |\boldsymbol{y}| = |\boldsymbol{z}|}{x.(\boldsymbol{y})P \mid x.(\nu \boldsymbol{v})[\boldsymbol{z}]P \rightarrow (\nu \boldsymbol{v})(P\{\boldsymbol{z}/\boldsymbol{y}\}|Q)}$$

$$\textsc{Par}\ \frac{P \rightarrow P'}{P|Q \rightarrow P'|Q} \qquad \textsc{Equiv}\ \frac{P \equiv P' \qquad P' \rightarrow Q' \qquad Q' \equiv Q}{P \rightarrow Q} \qquad \textsc{New}\ \frac{P \rightarrow P'}{(\nu\ x)\,P \rightarrow (\nu\ x)\,P'}$$

In closing this summary, we take the opportunity to observe that it is precisely the dynamics that differentiates this encoding. The equivalence that coincides with ambient isotopy is a *behavioral* equivalence, i.e. an equivalence of the dynamics of processes in the image of the encoding. In a marked departure from Gauss codes or DT-codes or Conway's "knotation" this facet of the encoding affords the *conflation* of notation scheme with invariant, providing a framework in which to establish the distinguishing power of the invariant and a language in which to express classes of knots as logical properties, as discussed below.

Characteristic formulae

Associated to the mobile process calculi are a family of logics known as the Hennessy-Milner logics. These logics typically enjoy a semantics interpreting formulae as sets of processes that when factored through the encoding outlined above allows an identification of classes of knots with logical formulae. In the context of this encoding the sub-family known as the spatial logics [11] [10] [10] are of particular interest providing several important features for expressing and reasoning about properties (i.e. classes) of knots.

structural connectives The spatial logics enjoy structural connectives corresponding, at the logical level, to the parallel composition $(P|Q)$ and new name $((\nu\ x)P)$ connectives for processes. As illustrated in the examples below, these connectives are extremely expressive given the shape of our encoding.

decideable satisfaction In [10] the satisfaction relation is shown to be decideable for a rich class of processes. It further turns out that the image of our encoding

is a proper subset of that class. This result provides the basis for an algorithm by which to search for knots enjoying a given property.

characteristic formulae In the same paper, Caires presents a means of calculating characteristic formulae, picking out equivalence classes of processes up to some limit on the support set of names. Composed with our encoding, this characteristic formula can be used to pick out characteristic formulae for knots.

Spatial logic formulae The grammar below (segmented for comprehension) summarizes the syntax of spatial logic formulae. We employ illustrative examples in the sequel to provide an intuitive understanding of their meaning referring the reader to [10] for a more detailed explication of the semantics.

BOOLEAN
$$A, B ::= T \mid \neg A \mid A \wedge B \mid \eta = \eta'$$

SPATIAL
$$\mid\ 0\ \mid A|B \mid x\circledR A \mid \forall x.A \mid Hx.A$$

BEHAVIORAL
$$\mid \alpha.A$$

RECURSION
$$\mid X(\boldsymbol{u}) \mid \mu X(\boldsymbol{u}).A$$

ACTION
$$\alpha ::= \langle x?(\boldsymbol{y})\rangle \mid \langle x!(\boldsymbol{y})\rangle \mid \langle \tau\rangle$$

NAME
$$\eta ::= x \mid \tau$$

Example formulae

Crossing as formula

$$\mathbf{C}(x0,x1,y0,y1,u) \triangleq \mu C(x0,x1,y0,y1,u).(\langle x0?(z)\rangle(\langle u!\rangle\langle y1!z\rangle C(x0,x1,y0,y1,u))$$
$$\wedge \langle y1?(z)\rangle(\langle u!\rangle\langle x0!z\rangle C(x0,x1,y0,y1,u))$$
$$\wedge \langle x1?(z)\rangle(\langle u?\rangle\langle y0!z\rangle C(x0,x1,y0,y1,u))$$
$$\wedge \langle y0?(z)\rangle(\langle u?\rangle\langle x1!z\rangle C(x0,x1,y0,y1,u)))$$

The lexicographical similarity between shape of this formula with the definition of the process representing a crossing reveals the intuitive meaning of this formulae. It describes the capabilities of a process that has the right to represent a crossing. What differentiates the formula from the process, however, is that the crossing process is the smallest candidate to satisfy the formulae. Infinitely many other processes – with internal behavior hidden behind this interface, so to speak – also satisfy this formulae. Even this simple formula, then, can be seen to open a new view onto knots, providing a computational interpretation of *virtual* knots in terms of simulation.

Note that this formula is derived by hand. A similar formula can be derived by employing Caires calculation of characteristic formula [10] to the process representing a crossing. In light of this discussion, in the subsequent examples we let

$[\![C]\!]_\phi(x0, x1, y0, y1, u)$ denote a formula specifying the dynamics we wish to capture of a crossing. To guarantee we preserve the shape of the interface and minimal semantics we demand that $[\![C]\!]_\phi(x0, x1, y0, y1, u) \Rightarrow \mathbf{C}(x0, x1, y0, y1, u)$.

Crossing number constraints Given a formula, $[\![C]\!]_\phi(x0, x1, y0, y1, u)$, as above we can use the structural connectives to specify constraints on crossing numbers, such as at least n crossings, or exactly n crossings.

$$\text{AT-LEAST-N}$$
$$K_\phi^{\geq n}(\boldsymbol{xs}, \boldsymbol{ys}) := \mathit{\Pi}_{i=0}^{n-1} Hu.[\![C]\!]_\phi(xs_i, ys_i, u)|T$$

$$\text{EXACTLY-N}$$
$$K_\phi^{=n}(\boldsymbol{xs}, \boldsymbol{ys}) := \mathit{\Pi}_{i=0}^{n-1} Hu.[\![C]\!]_\phi(xs_i, ys_i, u)|\neg(\forall x_0, x_1, y_0, y_1, u.[\![C]\!]_\phi(x_0, x_1, y_0, y_1, u)|T)$$

To round out this section, recall that the encoding of an n-crossing knot decomposes into a parallel composition of n *copies* of a crossing process together with a wiring harness. To specify different knot classes with the same crossing number amounts to specifying logical constraints on the wiring harness. In the interest of space, we defer examples to a forthcoming paper.

The proposed work and plan

Three aims inform the investigation proposed here:

Dynamics and invariants. Extend the well-established paradigm of associating algebras to topological spaces to the setting where the algebras hosting such invariants have dynamics in the sense that they explicitly represent or embed a model of computation;

Compositionality. Exploit compositionality to expose those features of a space that may be analyzed locally (like the polarity of crossing in a knot diagram) versus those that require global information (like the orientation of a knot);

Bisimulation. Establish a framework in which the proof principle and attendant proof methods of bisimulation may be exported to algebraic topology.

Work to be undertaken

The work to achieve these aims falls into three categories:

Theory. Establishing the encoding is really the first step. A great deal of work still remains in understanding the encoding via a principled investigation of its connections to existing work and limits of applicability.

Connection to other invariants. Current investigation suggests a natural correspondence to other invariants. The compositional nature of the encoding suggests a straightforward algorithm for calculating the Kauffman bracket from the process representation. This development warrants further research.

Knot classes as formulae. Of special interest is tabulating formulae identifying a wide variety of knot classes.

Braids and links. Further, on the knot side, because of the close correspondence of the basic crossing process to Conway's knotation primitive, it is natural to extend the encoding to links and braids. Initial work on the extension of the knot encoding to an encoding of links indicates a technical issue to resolve in order to implement this on the server.

Additional process calculi machinery. On the process calculi side it is natural to investigate other process calculi (e.g. ambient calculi [14]) as possible targets. Also, Gardner et al.'s context logic [23] seems particularly interesting as a language of properties of knots, braids and links.

Practice. Developing the encoding to the level of robustness that a knot database server can be made widely accessible will also test the ideas and provide a potentially interesting testbed for a number of practical applications from biology and other physical sciences. This work further divides into

Efficient encoding. To make the a practical and usable knot-search application it is necessary to factor the process-set semantics of the logic through an existing query engine. The proposal is to develop and prove correct an interpretation of the process-set semantics via XQuery, the XML query language. This interpretation will provide the core computation of the knot server.

Implementation. Once developed the XQuery semantics of process-sets will be implemented in an XML-aware version of the functional language, OCaml, known as OCamlduce and a webservice frontend developed for handling web-based queries. Additionally, integration with other knot packages, especially Scharein's knotplot [56] will provide rendering of search results as graphics. Finally, a frontend domain specific language for expressing queries in terms of partial specifications of knots will be derived from the spatial logic interpretation coupled with the XQuery semantics.

Communication. Each of these developments must be communicated back to the community. Being a multi-disciplinary effort the results span different communities (knot theorists and process algebraist) and as such communicating the results requires more effort because they must be couched in the technical languages of each field. Each theoretical investigation needs to be written up and published in peer-reviewed journals, such [JKTR] or [MSCS]. Additionally, the results of the

practical application is also of interest to the XML communities and emerging webservices communities.

General plan of work

The process by which the above objectives will be reached is now described.

The work will be carried out over a 3 year period. The project will begin 1 September 2007 and terminate on 31 August 2010, with deliverables expected at the end of each year.

Year 1. XQuery process-set semantics. Complete specification with proof of correctness available as technical report.

Tabulation of knot-class formulae. Complete specification of a number of distinguishing knot formulae, e.g. formulae for alternating knots, rational knots, toroidal knots, etc. available as technical report.

Correspondence to other invariants. Complete specification and proof of correctness of the calculation of the Kauffman bracket from the process encodings and characteristic formulae available as technical report.

Comparative study of context and spatial logics. Initial study of context logics versus spatial logics as a language for expressing properties of knots, links and braids.

Year 2. Knotserver DB: Implementation of XQuery semantics. Working implementation of the XQuery interpretation of process-set semantics in OCamlduce.

Knotserver DB: Knot properties. Working implementation of domain-specific language for expressing knot properties.

Knotserver DB: Webservice. Working implementation webservice frontend to knot database.

Knotserver DB: Sample formulae. Working implementation of knot formulae identified in first yeear expressed in domain specific knot property language.

Extended encoding to braids and links. Complete specification of the extension of the encoding to links and braids with proof of theorem analogous to main theorem of [32].

Year 3. XQuery process-set semantics. Conference submission of XQuery process-set semantics.

Knotserver DB: experience. Conference submission of experience paper on the construction of the knotserver db.

Tabulation of knot-class formulae. Conference submission on the specification of knot properties.

Correspondence to other invariants. Journal submission on the correspondence of this encoding to other knot invariants.

Comparative study of context and spatial logics. Conference submission on study of context logics versus spatial logics as a language for expressing properties of knots, links and braids.

Extended encoding to braids and links. Journal submission on the extension of the encoding to links and braids with proof of theorem analogous to main theorem of [32].

General personnel effort

The PI will be devoting 25% of the academic year and 100% (of two months) of each of the two summers to this project. The co-PI will be devoting 35% of each year to the project; the co-PI will perform the majority of his work at his home site, Biosimilarity LLC in Seattle, and will visit the Texas State site once per quarter during the project. The co-PI will visit near the beginning of the project to interact with the students. The PI and co-PI have successfully worked together at-a-distance using a combination of e-mail, internet messaging, and internet telephony; they will continue to do so during this project as well, at least twice weekly.

A graduate student at the Master's level from Mathematics, will be recruited or hired at the beginning of the project, with the aim that they come from an underrepresented group. The graduate assistant will be at 50% effort. Likewise, a junior member of Biosimilarity staff will be devoting 50% of their time on the project for technical support and development activities.

The high school students will attend MathWorks from late June to late July. The co-PI will come for an extended visit during this time to familiarize the Mathworks team with the process logics.

Equipment and materials

A MacBook Pro and a MacPro is requested for the Texas State site and two MacBook Pro's for the Biosimilarity site. The co-PI will be in charge of directing and managing the software development. The two nodes are needed for the co-PI to have the specific machine architecture envisioned for serving the knots and braids component. Instructional materials for the students will be purchased as well.

Background for the educational component

One Mathematics graduate student will participate in the project. Ethnic minorities form 25% of Texas State's student body. Texas State is one of the top 20 producers of Hispanic baccalaureate graduates in the nation. Its Mathematics Department has a strong tradition of supporting and engaging students from underrepresented populations. Moreover, the PI has a proven record of accomplishment in engaging students in research projects appropriate to their level of development.

The project will also engage a team of 2-3 high school students from the Texas Mathworks summer program. Texas Mathworks conducts a 6-week Honors Summer Math Camp (HSMC) for highly talented high school students. First year students take courses in Number Theory, Mathematica Lab, Problem Solving and an Honors Seminar. Returning students take courses in Abstract Algebra, Analysis, and Knot Theory. Returning students also work on research projects, mentored by a faculty member. These projects have been outstanding, resulting in 43 Siemens-Westinghouse semi-finalists, 21 regional finalists, and 6 national finalists (2 teams) in the past 5 years. The PI will mentor such a team, with the co-PI lending his expertise on process calculi and Xquery (the standard extensible query language).

References

1.
2.
3. Martín Abadi and Bruno Blanchet. Analyzing security protocols with secrecy types and logic programs. In *POPL*, pages 33–44, 2002.
4. Martín Abadi and Bruno Blanchet. Secrecy types for asymmetric communication. *Theor. Comput. Sci.*, 3(298):387–415, 2003.
5. Samson Abramsky. Algorithmic game semantics and static analysis. In *SAS*, page 1, 2005.
6. P. D. Bangert. *Algorithmic Problems in the Braid Groups*. PhD thesis, University College London, 2002.
7. Dror Bar-Natan. The knot atlas.
8. Dror Bar-Natan. Knot-theory, the mathematica package.
9. S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *J. Assoc. Comput. Mach.*, 31(3):560–599, 1984.
10. Luís Caires. Behavioral and spatial observations in a logic for the pi-calculus. In *FoSSaCS*, pages 72–89, 2004.
11. Luís Caires and Luca Cardelli. A spatial logic for concurrency (part i). *Inf. Comput.*, 186(2):194–235, 2003.
12. Luís Caires and Luca Cardelli. A spatial logic for concurrency - ii. *Theor. Comput. Sci.*, 322(3):517–565, 2004.
13. Jorge Alberto Calvo. Knot enumeration through flypes and twisted splices. *J. Knot Theory Ramifications*, 6(6):785–798, 1997.
14. Luca Cardelli. Brane calculi. In *CMSB*, pages 257–278, 2004.
15. A. Caudron. Classification des noeuds et des enlacements. Prepublication Math. d'Orsay, Orsay, France: Universite Paris-Sud, 1981.
16. J. H. Conway. An enumeration of knots and links, and some of their algebraic properties. In J. Leech, editor, *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, pages 329–358. Pergamon, Oxford, 1970.
17. Vincent Danos and Cosimo Laneve. Core formal molecular biology. In Pierpaolo Degano, editor, *ESOP*, volume 2618 of *Lecture Notes in Computer Science*, pages 302–318. Springer, 2003.
18. C. H. Dowker and Morwen B. Thistlethwaite. Classification of knot projections. *Topology Appl.*, 16(1):19–31, 1983.
19. Wan Fokkink, Rob van Glabbeek, and Paulien de Wind. Compositionality of Hennessy-Milner logic through structural operational semantics. In *Fundamentals of computation theory*, volume 2751 of *Lecture Notes in Comput. Sci.*, pages 412–422. Springer, Berlin, 2003.
20. Cédric Fournet, Fabrice Le Fessant, Luc Maranget, and Alan Schmitt. Jocaml: A language for concurrent distributed and mobile programming. In *Advanced Functional Programming*, pages 129–158, 2002.
21. Cédric Fournet, Georges Gontheir, Jean-Jacques Lévy, Luc Maranget, and Didier Rémy. A calculus of mobile agents. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR 1996*, volume 1119 of *Lecture Notes in Computer Science*, pages 406–421. Springer-Verlag, 1996.
22. Rūsiņš Freivalds. Knot theory, Jones polynomial and quantum computing. In *Mathematical foundations of computer science 2005*, volume 3618 of *Lecture Notes in Comput. Sci.*, pages 15–25. Springer, Berlin, 2005.
23. Philippa Gardner and Sergio Maffeis. Modelling dynamic web data.
24. H. Gruber. http://home.in.tum.de/ gruberh/.
25. C. A. R. Hoare. *Communicating sequential processes*. Prentice Hall International Series in Computer Science. Prentice Hall International, Englewood Cliffs, NJ, 1985. With a foreword by Edsger W. Dijkstra.

26. C. A. R. Hoare. Notes on communicating sequential systems. In *Control flow and data flow: concepts of distributed programming (Marktoberdorf, 1984)*, volume 14 of *NATO Adv. Sci. Inst. Ser. F Comput. Systems Sci.*, pages 123–204. Springer, Berlin, 1985.
27. C. A. R. Hoare. Algebraic specifications and proofs for communicating sequential processes. In *Logic of programming and calculi of discrete design (Marktoberdorf, 1986)*, volume 36 of *NATO Adv. Sci. Inst. Ser. F Comput. Systems Sci.*, pages 277–300. Springer, Berlin, 1987.
28. Jim Hoste. The enumeration and classification of knots and links. In *Handbook of knot theory*, pages 209–232. Elsevier B. V., Amsterdam, 2005.
29. Noriko Imafuji and Mitsuyuki Ochiai. Computer aided knot theory using Mathematica and MathLink. *J. Knot Theory Ramifications*, 11(6):945–954, 2002. Knots 2000 Korea, Vol. 3 (Yongpyong).
30. Allen L. Brown Jr., Cosimo Laneve, and L. Gregory Meredith. Piduce: A process calculus with native xml datatypes. In Mario Bravetti, Leïla Kloul, and Gianluigi Zavattaro, editors, *EPEW/WS-FM*, volume 3670 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2005.
31. T. P. Kirkman. The enumeration, description, and construction of knots of fewer than ten crossings. *Trans. Roy. Soc. Edin.*, 32:281–309, 1885.
32. D. F. Snyder L. G. Meredith. Knots as processes. Pre-print, November 2006.
33. Cosimo Laneve and Gianluigi Zavattaro. Foundations of web transactions. In *FoSSaCS*, pages 282–298, 2005.
34. Tomáš Liko and Louis H. Kauffman. Knot theory and a physical state of quantum gravity. *Classical Quantum Gravity*, 23(4):R63–R90, 2006.
35. C. N. Little. On knots, with a census to order 10. *Trans. Conn. Acad. Sci.*, 18:374–378, 1885.
36. C. N. Little. Alternate ± knots of order 11. *Trans. Roy. Soc. Edin.*, 36:253–255, 1890.
37. C. N. Little. Non-alternate ± knots. *Trans. Roy. Soc. Edin.*, 39:771–778, 1900.
38. Charles Livingston. http://www.indiana.edu/ knotinfo/.
39. Greg Meredith and Steve Bjorg. Contracts and types. *Commun. ACM*, 46(10):41–47, 2003.
40. L. Gregory Meredith and Matthias Radestock. A reflective higher-order calculus. *Electr. Notes Theor. Comput. Sci.*, 141(5):49–67, 2005.
41. L.G. Meredith and Matthias Radestock. A reflective higher-order calculus. In Mirko Viroli, editor, *ETAPS 2005 Satellites*. Springer-Verlag, 2005.
42. Robin Milner. *A Calculus of Communicating Systems*. Springer Verlag, 1980.
43. Robin Milner. Elements of interaction - turing award lecture. *Commun. ACM*, 36(1):78–89, 1993.
44. Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes. I, II. *Inform. and Comput.*, 100(1):1–40, 41–77, 1992.
45. Robin Milner and Davide Sangiorgi. Barbed bisimulation. In Werner Kuich, editor, *ICALP*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695. Springer, 1992.
46. Corrado Priami, Aviv Regev, Ehud Y. Shapiro, and William Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Inf. Process. Lett.*, 80(1):25–31, 2001.
47. Stuart Rankin, Ortho Flint, Ralph Furmaniak, and Bruce Fontaine. Knotilus.
48. Stuart Rankin, Ortho Flint, and John Schermann. Enumerating the prime alternating knots. I. *J. Knot Theory Ramifications*, 13(1):57–100, 2004.
49. Stuart Rankin, Ortho Flint, and John Schermann. Enumerating the prime alternating knots. II. *J. Knot Theory Ramifications*, 13(1):101–149, 2004.
50. Amitai Regev and Ehud Y. Shapiro. Cells as computation. In Corrado Priami, editor, *CMSB*, volume 2602 of *Lecture Notes in Computer Science*, pages 1–3. Springer, 2003.
51. Kurt Reidemeister. Knoten und Verkettungen. *Math. Z.*, 29(1):713–729, 1929.
52. R. Sazdanovic S. Jablan. Linknot.
53. David Sangiorgi and David Walker. *The π-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.
54. Davide Sangiorgi. On the proof method for bisimulation (extended abstract). In Jirí Wiedermann and Petr Hájek, editors, *MFCS*, volume 969 of *Lecture Notes in Computer Science*, pages 479–488. Springer, 1995.

55. Davide Sangiorgi. Bisimulation in higher-order process calculi. *Information and Computation*, 131:141–178, 1996.
56. R. Scharein. http://www.pims.math.ca/knotplot/.
57. Robert G. Scharein. *Interactive Topological Drawing*. PhD thesis, Department of Computer Science, The University of British Columbia, 1998.
58. P.G. Tait. On knots, i, ii, iii. In *Scientific Papers*, volume 1, pages 273–347. Cambridge University Press, Cambridge, 1898.
59. Pawel T. Wojciechowski and Peter Sewell. Nomadic pict: Language and infrastructure design for mobile agents. In *ASA/MA*, pages 2–12, 1999.