Chu Spaces and their Interpretation as Concurrent Objects

Vaughan Pratt*
Dept. of Computer Science
Stanford University
Stanford, CA 94305-2140
pratt@cs.stanford.edu

January 9, 2005

Abstract

A Chu space is a binary relation $=\!|$ from a set A to an antiset X defined as a set which transforms via converse functions. Chu spaces admit a great many interpretations by virtue of realizing all small concrete categories and most large ones arising in mathematical and computational practice. Of particular interest for computer science is their interpretation as computational processes, which takes A to be a schedule of events distributed in time, X to be an automaton of states forming an information system in the sense of Scott, and the pairs (a, x) in the $=\!|$ relation to be the individual transcriptions of the making of history. The traditional homogeneous binary relations of transition on X and precedence on A are recovered as respectively the right and left residuals of the heterogeneous binary relation $=\!|$ with itself. The natural algebra of Chu spaces is that of linear logic, made a process algebra by the process interpretation.

1 Introduction

Two pressing questions for computer science today are, what is concurrency, and what is an object? The first question is of interest to today's sibling growth industries of parallel computing and networks, both of which stretch our extant models of computation well beyond their sequential origins. The second is relevant to programming languages, where the definition of object seems to be based more on whatever software engineering methodologies happen to be in vogue than on the intuitive sense of "object."

One recent view of computation [Tra95] classifies the extant models under the three headings of logic, networks, and automata. But while this perspective

*This work was supported by ONR under grant number N00014-92-J-1974

nicely ties together three long-established computational frameworks, it neglects more recent developments such as the recent work of Abramsky on interaction categories as exemplified by the category *SProc* [GN95], and of Milner on the π-calculus [MPW92] and more recently action calculi [Mil93]. Moreover its conception of network is more the channel-connected modules of Kahn [Kah74] than the alternating places and transitions of Petri [Pet62].

Petri nets express the duality of events and states in terms of a "token game" played on a bipartite graph. This bipartiteness is the distinguishing feature of Petri nets, resulting in fine-grained or move-based execution where tokens move alternately between events or transitions and states or places. This is in contrast to the coarse-grained or ply-based execution of transition systems where edges connect states, and schedules where edges connect events.

A popular strategy for controlling the conceptual complexity of large concurrent systems is to decompose them into modules that assemble or compose in mathematically formalizable ways. The basic tool for formalizing the composition of modules is algebra. This motivates the development of process algebras forming a suitable basis for concurrent programming languages. Noteworthy such algebras include Hoare's Communicating Sequential Processes (CSP) [Hoa78], Milner's Calculus of Communicating Systems (CCS) [Mil89], and Bergstra and Klop's Algebra of Concurrent Processes (ACP) [BK84, BK89].

The token game does not lend itself well to algebra, a limitation addressed by Nielsen et al [NPW81] with the notion of event structure, which they show to be dual to prime algebraic domains, one kind of Scott domain [Sco76]. This duality is an instance of Stone duality [Sto36], a subject that has matured considerably in half a century [Joh82].

We see the essential elements of concurrency as residing in the topics of Petri nets, event structures, domains, Stone duality, and process algebra.

As the separate chapters of a theory of concurrency, these topics make concurrency something of a jigsaw puzzle. One is then naturally led to ask whether the pieces of this puzzle can be arranged in some recognizable order.

In this paper we review the outcome to date of our recent investigations of Chu spaces as a candidate unifying framework for these aspects of concurrency [Pra92, Pra93, Gup93, GP93, Gup94, Pra94b, Pra95a]. Their simplicity is deceptive, and two years of experience with them have convinced us that they are more than adequately equipped for this role.

Chu spaces are simple, useful, and well-organized. With regard to simplicity, a Chu space is merely a matrix that transforms by deleting and copying columns (antimapping) and identifying and adjoining rows (ordinary mapping). The next section defines this process more precisely in terms of a converse function or antifunction g specifying where the surviving columns were copied from and a forward function f specifying where the resulting rows are to be sent to. These contravariant column manipulations constitute mental preparation of states while the covariant row manipulations constitute physical mapping of points; thus Chu spaces transform by looking before they leap.

With regard to utility, besides the motivating application to concurrent computation, Chu spaces find application in mathematics, where they organize re-

lational structures, topology, and duality into a unified framework [Pra95b]; physics, where they provide a process interpretation of wavefunctions [Pra94a]; and philosophy, where they offer a solution to Descartes' problem of the mechanism by which the mind interacts with the body [Pra95a]. Common to these applications is the construction of "everything" in the domain in question in terms of the interaction of appropriate polar opposites in the domain: in mathematics as the interaction of sets and antisets, in physics as the interaction of particles and waves, in philosophy as the interaction of body and mind. It is a measure of the robustness of the notion of Chu space that none of these applications call for any adjustment to its definition.

With regard to organization, Chu spaces and their transforms form a remarkably well-endowed category, being concrete and coconcrete, self-dual, bicomplete, and symmetric monoidal closed. These properties are expressible concretely as operations on the category corresponding to the operations of linear logic and bringing it as close to being a model of full linear logic as any comparably simple structure. The process interpretation of Chu spaces makes this structure a process algebra interpretation of linear logic. This reinforces the connections previously noticed by Asperti [Asp87] and Gehlot and Gunter [GG89] between linear logic and concurrency in the Petri net setting (Section 4).

2 Definitions

Definition 1 A *Chu space over* a set K is a triple $\mathcal{A} = (A, \rightleftharpoons, X)$ consisting of sets A and X and a function $\rightleftharpoons : A \times X \to K$, that is, an $A \times X$ matrix whose elements are drawn from K. We write the entry at (a, x) as either $\rightleftharpoons(a, x)$ or $a \rightleftharpoons x$. ∎

We may view $\mathcal{A}$ as being organized either into rows or columns. When viewing $\mathcal{A}$ by row we regard A as the carrier of a structure and the row indexed by a as the complete description of element a. The *description function* $\widetilde{\rightleftharpoons} : A \to K^X$ is the function satisfying $\widetilde{\rightleftharpoons}(a)(x) = \rightleftharpoons(a, x)$, and assigns to each a its description. We write $\widetilde{a}$ for the description of a, namely $\widetilde{\rightleftharpoons}(a)$, and $\widetilde{A}$ for the set $\{\widetilde{a} \mid a \in A\}$ of all rows of $\mathcal{A}$. When the description function is injective (no repeated rows) we say that $\mathcal{A}$ is T_0 (or coextensional).

When viewing $\mathcal{A}$ by column we view A as consisting of locations or variables with values ranging over K, and the column indexed by x as one of the permitted assignments to these variables. The *extension function* $\widetilde{\models} : X \to K^A$ satisfies $\widetilde{\models}(x)(a) = \models(x, a)$; a state $x \in X$ is understood as merely the *name* of an assignment or binding of values to variables. The notations $\widetilde{x}$ and $\widetilde{X}$ are the evident duals of $\widetilde{a}$ and $\widetilde{A}$. When the extension function is injective (no repeated columns) we call $\mathcal{A}$ *extensional*. In a nonextensional Chu space two states may name the same assignment.

A Chu space can in this way be seen as a multiset A of rows from K^X and a multiset X of columns from K^A, with "multiset" replaced by "set" when

T_0 and extensional. The rows present the physical, concrete, conjunctive, or yang aspects of the space, while the columns present the mental, coconcrete, disjunctive, or yin aspects. We may regard rows and columns as characteristic functions of subsets of respectively X and A, where K is taken to consist of the degrees of membership, with $K = 2 = \{0, 1\}$ giving the ordinary notion of membership, 1 for in and 0 for out.

The Chu space obtained by identifying all pairs x, y of column indices for which $\widetilde{x} = \widetilde{y}$, and likewise for row indices, is called the *skeleton* of that Chu space. A *normal* Chu space is one satisfying $\widetilde{x} = x$ for all $x \in X$ (whence $\widetilde{X} = X$), and can be written as simply (A, X), $\models\!\!|(a, x)$ being definable as $x(a)$. A normal space is automatically extensional but need not be T_0.

The *dual* of Chu space $(A, \models\!\!|, X)$ is its transpose, notated $(X, \models, A)$ where $\models(x, a) = \models\!\!|(a, x)$. A *conormal* space is the dual of a normal space, automatically T_0 but not necessarily extensional.

Chu transforms. Just as vector spaces transform via linear transformations and posets via monotone functions, so do Chu spaces transform via Chu transforms, turning the class of Chu spaces into the category $\mathbf{Chu}(\mathbf{Set}, K)$ or just $\mathbf{Chu}_K$.

Definition 2 Given source and target Chu spaces $\mathcal{A} = (A, \models\!\!|, X)$ and $\mathcal{A}' = (A', \models\!\!|', X')$, a *Chu transform* $(f, g) : \mathcal{A} \to \mathcal{A}'$ consists of a pair of functions $f : A \to A'$, $g : X' \to X$ satisfying the following *adjointness* condition.

$$\forall a{\in}A\ \forall x'{\in}X'\ .\ f(a) \models\!\!|'\, x' \ =\ a \models\!\!| g(x').$$

∎

Chu transforms may be understood operationally as "look before you leap," that is, mental preparation followed by physical transformation, with an intermediate Chu space $\mathcal{A}'' = (A, \models\!\!|'', X')$ representing the result of just the mental preparation. The columns of $\mathcal{A}''$ all appear in $\mathcal{A}$ while its rows all appear in $\mathcal{A}'$. Column x' of $\mathcal{A}''$ appears as column $g(x')$ of $\mathcal{A}$ and row a of $\mathcal{A}''$ appears as row $f(a)$ of $\mathcal{A}'$. Either one of these requirements suffice to determine $\models\!\!|''(a, x')$ uniquely, as either $\models\!\!|(a, g(x'))$ or $\models\!\!|'(f(a), x')$. The adjointness condition is the necessary and sufficient condition for their consistency. This process is illustrated by the following example (which incidentally realizes the projection of the two dimensional vector space over $GF(2)$ onto one axis).

$$\begin{array}{cccc} 0&0&0&0\\ 0&1&0&1\\ 0&0&1&1\\ 0&1&1&0 \end{array} \quad \begin{array}{c}\text{delete cols 2,3}\\ \longrightarrow \\ \text{copy cols 0,1}\end{array} \quad \begin{array}{cccc} 0&0&0&0\\ 0&1&0&1\\ 0&0&0&0\\ 0&1&0&1 \end{array} \quad \begin{array}{c}\text{send rows 2,3}\\ \text{to rows 0,1}\\ \longrightarrow \\ \text{adjoin original}\\ \text{rows 2 and 3}\end{array} \quad \begin{array}{cccc} 0&0&0&0\\ 0&1&0&1\\ 0&0&1&1\\ 0&1&1&0 \end{array}$$

Columns 2 and 3 are deleted, and a copy made of columns 0 and 1. Then row 0 is identified with 2 and 1 with 3, and two new rows adjoined (which happen to be rows 2 and 3 of the source but they could have been anything had we

not cared whether the target was a vector space). (The intermediate space is not a vector space, but would have been the one-dimensional vector space had we done the row identifying in the first stage and the column copying in the second, a reasonable order.)

The "mental preparation" step therefore consists of deleting some columns and making copies of others. *No new columns are introduced*, so no structure (in the form of disallowed states) is lost. If the target has structure absent from the source, the structure must first be added by deleting columns or the row mapping will not be possible. This gives Chu transforms the character of structure-preserving homomorphisms [Pra95b], with continuous functions then falling out as an obvious special case when analyzed as above. All relational structures are realizable as Chu spaces [Pra93, Pra95b], as are topological spaces [LS91]. These representations can be combined to represent topological relational structures such as topological groups, topological vector spaces (the main *-autonomous category studied in [Bar79]), ordered Stone spaces [Pri70], and so on. More recently (note in preparation) we have shown that every small concrete category C is realizable in $\mathbf{Chu}_K$ where K is the disjoint union of the underlying sets of the objects of C.

3 Process

3.1 Interaction of events and states

This section gives the computational or process interpretation of Chu spaces, in terms of their internal row-and-column structure rather than their external linear-logic structure, the topic of the next section.

We propose to understand computation in terms of the interaction of a pure schedule, understood as a set of concurrent events, with a pure automaton, understood as a set of possible alternative worlds or states, This parallels our proposed interpretations of other phenomena in terms of the interaction of opposites appropriate to those phenomena.

Although computation has a number of facets, a particularly problematic one is concurrency, the search for whose essence led us to the Chu space model. We begin by discussing what is problematic about concurrency.

To cater for concurrency we moved in the 1980's from the traditional state-oriented view of computation that we had embraced in the 1970's to a more event-oriented view. The former understands concurrency in terms of interleaving: concurrent events are those that can happen *in either order*. The latter understands concurrency in terms of independence or asynchrony: concurrent events are those that bear no temporal relationship to each other, happening *without regard for order* as distinct from simultaneous or synchronized events.

The main difference here is that the state oriented view cannot distinguish between mutual exclusion and independence for events that we wish or need to view as atomic. For example, if two children each wish to ride a pony, they are much happier when there are two ponies. One pony offers the choice of two

possible sequential behaviors ab or ba. Two ponies dispense with both choice and delay by permitting the single concurrent behavior $a\|b$. Notice that our discussion of the situation is in terms of the rides as atomic actions; we did not say what goes on during the ride in drawing this distinction. The traditional state model cannot draw this distinction because it automatically identifies $a\|b$ and $ab + ba$ when the rides a and b are understood as atomic.

By expressing a behavior as a partially ordered set of events, and a process as a set of possible behaviors, this distinction can now be drawn naturally by representing $a\|b$ as an unordered set of two events constituting one behavior, and $ab + ba$ as two possible behaviors, one for each linear ordering of the two events.

In practice however, people naturally think as much in terms of states as events. What is really needed is a simple connection between the two viewpoints facilitating a smooth passage between them.

Chu spaces address concurrency by admitting their interpretation as processes. The rows and columns of a Chu space are interpreted as respectively the events and states of the process, while the entries of the matrix are interpreted as the interaction of event and state, more specifically as the recording, by the state, of information about the event. Events are regarded as points in time while states are points in information space. Events specify no particular information, though they are associated with incremental changes to information, just as derivatives are associated with no particular position but rather only with incremental changes to position. Dually, states do not exist at any particular moment in time, but rather are associated with incremental changes in time: time passes while one waits in a state, during which (at least for discrete time) no events change status. We distinguish between time as the actual position of events and information as the *knowledge* of those positions.

A Chu space as a process may be viewed either imperatively or declaratively by focusing on the columns or the rows respectively of the representing space.

3.2 Extracting transitions and constraints

The Chu space model of behavior may be usefully contrasted with the traditional imperative transition-based approach as well as with the declarative approach involving temporal (precedence, delay, etc.) constraints between events, as follows (we assume $K = 2$ for simplicity).

The basic component of the imperative approach is the transition, consisting of a pair (x, y) of states whose meaning is that it is *possible* to pass from state x to state y. An automaton is a set of such transitions, which if acyclic may be reflexively and transitively closed to partially order the set X of states.

The corresponding basic component in the event-oriented approach is the precedence constraint, consisting of a pair (a, b) of events whose meaning is that it is *necessary* to perform event a before event b. A schedule is a set of such constraints, acyclic when there is no deadlock and hence transitively closable to partially order the set A of events.

Both these views involve *ongoing motion* between points of the same type, namely states in an information space in the imperative case and events in time in the declarative case. These motions differ with regard to the interpretation of branching, which is understood disjunctively in an automaton (only one path is taken) but conjunctively in a schedule (all constraints must be observed).

The Chu space view involves not ongoing or connected motion but instead recording. A *recording* is a pair (a, x) consisting of an event a and a state x. A Chu space is a set of such recordings. This makes it a bipartite directed graph, namely a binary relation from the set A of events to the set X of states, for which "acyclic" and "transitive" are not usefully definable. In this way Chu spaces capture the fine-grained move-orientation of Petri nets, as opposed to the traditional coarse-grained ply-orientation of transitions and precedence constraints, but with more satisfactory algebra than Petri nets.

What *is* definable for any two binary relations with a common source A is their *right residual*. If $R \subseteq A \times B$ and $T \subseteq A \times C$ then their *right residual* $R \backslash T$, as a relation from B to C, is defined as consisting of those pairs (b, c) such that for all $a \in A$, aRb implies aTc.

Now the right residual $\models\backslash\models$ of the Chu space relation $\models$ with itself yields a binary relation on X. This relation is the largest possible transition relation on X having the property that no transition from x to y can "undo" an event that is recorded in state x as having already happened. It is also the natural "inclusion" or bitwise order on columns, in which $0 \leq 1$, which of course makes it clear that it partially orders X.

The left residual of two relations with a common target is defined dually. If $S \subseteq B \times C$ and $T \subseteq A \times C$, then their *left residual* T/S, as a binary relation from A to B, is defined as consisting of those pairs (a, b) such that for all $c \in A$, bSc implies aTc. The left residual $\models/\models$ yields a binary relation on A, namely the largest possible precedence relation with the property that for every pair (a, b) in the relation, no state has witnessed b without a. This too is a partial order.

The residual-based derivations of ongoing state-state and event-event *motion* in terms of event-state *recording* lifts to more general relationships than the all-or-nothing ($K = 2$) one of possibility of transition and necessity of precedence, a mathematically rich and interesting topic that space and time prevent us from delving into here. To summarize briefly, distances are organized as a *quantale* [Ros90] or complete semilattice monoid. This combines our earlier investigations of purely temporal structures [CCMP91] with our more recent work on time-information duality and Chu spaces in a way that permits the temporal-structure results to be lifted directly, via duality, to information structures such as automata and Scott-style information systems. This generalizes the two-valued generalized metric associated with partial orders to other metrics such as causal-accidental order, strict-nonstrict order, and various notions of real time (*op.cit.* p.187).

3.3 Interpretation as Objects

As an unintended bonus, Chu spaces admit a straightforward interpretation as concurrent objects. Whereas statically sets and antisets are indistinguishable, dynamically they are distinguished by transforming respectively via functions and antifunctions. Noting that the set of states of an automaton forms an antiset, understood as a mental entity, we take an object to be a set, understood as physical. As a conjunctive set, the points of an object coexist, in contrast to a disjunctive antiset of states, the sense in which we consider an object to be concurrent. A collection $\mathcal{A}_1, \ldots, \mathcal{A}_n$ of objects is concurrent in the sense that its sum or coproduct $\sum_i \mathcal{A}_i$ is a concurrent object. In the absence of all constraints (i.e. the set of states of $\mathcal{A}$ is K^A), the object is truly concurrent, consisting of independent events. The imposition of constraints creates interferences between events such as conflicts and precedence constraints.

Although the methods of a given class are traditionally viewed as functions performable by objects of that class, in the context of Chu spaces we propose to take as the methods of a Chu space its states, which are just the permitted assignments of values from K to the physical points or locations of the space.

4 Algebra

The previous section examined Chu spaces from the inside in terms of their components. In this section we consider them from the outside, in terms of their language or algebra. The natural language of Chu spaces is linear logic, which appears to us to be as fundamental as any other process language, with Chu spaces giving it a sensible and useful interpretation.

4.1 Chu space interpretation of linear logic

As with Boolean logic there are various choices of basis for the set of all linear logic operations (though none as small as Boolean logic's single NAND operation!). A particularly natural basis, both in its own right and for our process interpretation, consists of four operations and two constants: the additive disjunction *plus*, $\mathcal{A} \oplus \mathcal{B}$, the multiplicative conjunction (tensor product) *times*, $\mathcal{A} \otimes \mathcal{B}$, linear negation $\mathcal{A}^\perp$, and the exponential or modality *of-course*, $!A$. Associated with plus and times are their respective units or zeroary operations 0 and 1.

When negation is omitted from this list, and $!\mathcal{A}$ is interpreted as $\mathcal{A}$, the laws of linear logic expressible in the remaining language are merely those of ordinary number theory. Larry Moss has pointed out to us that this remains true when $\mathcal{A}^\ddagger = (!\mathcal{A})^\perp$ is taken as basic in place of $!\mathcal{A}$ and interpreted numerically as 2^A (motivations: the axiom $(\mathcal{A} \oplus \mathcal{B})^\ddagger = \mathcal{A}^\ddagger \otimes \mathcal{B}^\ddagger$ can be expressed without negation [Pra93], and the 2^A offers a rationale for Girard's "exponential").

The presence of negation adds a completely new dimension to number theory. To begin with, plus and times and of-course each has its own De Morgan dual, respectively the additive conjunction *with*, $\mathcal{A}\&\mathcal{B}$, the multiplicative disjunction

par, $\mathcal{A}⅋\mathcal{B}$, and the dual exponential *why-not* $?\mathcal{A}$, with the first two having respective units $\top$ and $\bot$. Linear implication $\mathcal{A}\multimap\mathcal{B}$ is defined as $(\mathcal{A}\otimes\mathcal{B}^\bot)^\bot$ or $\mathcal{A}^\bot⅋\mathcal{B}$, while intuitionistic implication $\mathcal{A}\Rightarrow\mathcal{B}$ is defined as $!\mathcal{A}\multimap\mathcal{B}$. None of these operations has a number-theoretic interpretation.

With negation present, the basic operations may be interpreted as acting on Chu spaces as follows. $\mathcal{A}^\bot$ is just the dual or transpose of $\mathcal{A}$, $!\mathcal{A}$ is the carrier A of $\mathcal{A}$ as a Chu space, namely the normal Chu space (A, K^A).

$\mathcal{A}\oplus\mathcal{B}$ for $\mathcal{A}=(A,\not=_A,X)$, $\mathcal{B}=(B,\not=_B,Y)$ is $(A+B,\not=,X\times Y)$ where $\not=$ is defined by $a\not=(x,y)=a\not=_A x$, $b\not=(x,y)=b\not=_B y$. Its unit is $(0,!,1)$.

With the same $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A}\otimes\mathcal{B}$ is $(A\times B,\not=,|\mathcal{A}\multimap\mathcal{B}^\bot|)$ whose states are all Chu transforms (f,g) from $\mathcal{A}$ to $\mathcal{B}^\bot$ (so $f:A\to Y$ and $g:B\to X$) and where $(a,b)\not=(f,g)$ is defined equivalently (by the adjointness condition) as either $b\not=_B f(a)$ or $a\not=_A g(b)$. Its unit is the $1\times K$ normal Chu space whose one row is the identity function on K.

This completes the Chu space interpretation of the basic operations and constants. The interpretations of the derived operations are obtained from their definitions.

4.2 Process interpretation of linear logic

The computational significance of the operations is as follows; we leave negation to the end.

$\mathcal{A}\oplus\mathcal{B}$ is the asynchronous (noncommunicating or parallel play) concurrent composition of $\mathcal{A}$ and $\mathcal{B}$. If $\mathcal{A}$ is the process of my eating dinner and I take 47 bites, and $\mathcal{B}$ is you taking 53 bites, then $\mathcal{A}\oplus\mathcal{B}$ is us eating dinner together taking 100 bites in silence, that is, with no interaction. Our possible joint states form the set $X\times Y$ of all possible pairs of our individual states. The associated unit 0 has no events and 1 state, and is as unconstrained as a zero-event process gets.

$\mathcal{A}\otimes\mathcal{B}$ is the orthocurrence [Pra85, Pra86] or interaction or flow-through of $\mathcal{A}$ and $\mathcal{B}$, as with three trains passing through two stations to yield six train-station events. Each event is of the form (a,b) pairing a train with a station. Each state is of the form (f,g) where $f:A\to Y$ specifies for each train a what state the stations collectively appear to be in as seen from that train, while $g:B\to X$ does the same for the trains collectively as seen from each station. The associated unit 1 has one event and is completely unconstrained in that it has all states possible for a one-event process, namely K. When $\mathcal{A}$ and $\mathcal{B}$ realize posets, $\mathcal{A}\otimes\mathcal{B}$ realizes the poset that is their direct product (same as their tensor product since **Pos** is cartesian closed).

$!\mathcal{A}$ consists of the events of $\mathcal{A}$ less their constraints, making it a pure set of independent events.

With the operations and constants covered so far (including $\mathcal{A}^\ddagger$), it is impossible to introduce constraints starting from only unconstrained processes. The only processes are then in effect cardinals, from which follows the completeness of number theory as an axiomatization of the basic operations without negation—any number theoretic counterexample to a nontheorem of number

theory becomes a process counterexample making it also a nontheorem of process algebra. (The only awkward part with $\mathcal{A}^\ddagger$ is to show completeness for number theory of the axiomatization $(\mathcal{A}+\mathcal{B})^\ddagger = A^\ddagger \times B^\ddagger$, $0^\ddagger = 1$, $1^\ddagger = 1+1$.)

Negation merely dualizes our point of view, interchanging the qualities of event and state. States become physical entities and events mental, as in an automaton or a computer program where the locations in the text are real and the runtime events must be imagined. Actually running the program undoes the negation; the program itself disappears from physical view, at least in principle, and the runtime events become physical.

Linear implication $\mathcal{A} \multimap \mathcal{B}$ can be understood as $\mathcal{B}$ observing $\mathcal{A}$, a point of view we first developed in [CCMP91]. The definition reveals observation to be equivalent to the interaction $\mathcal{A} \otimes B^\perp$ of the events of the observed $\mathcal{A}$ with the states of the observer $\mathcal{B}$, namely where $\mathcal{B}$ records the observations, with the individual recordings of event a of $\mathcal{A}$ in state y of $\mathcal{B}$ as the *recording events* of $\mathcal{A} \otimes \mathcal{B}^\perp$. Taking the dual of this then turns the recording events into the states of $\mathcal{A} \multimap \mathcal{B}$, which is how the recording events should appear to an observer of $\mathcal{B}$ observing $\mathcal{A}$.

5 Universality

The previous sections emphasized the computational or information-processing and behavioral or scheduling interpretation of Chu spaces, our original motivation bringing us to them. In this section we review more briefly three other areas in which Chu spaces seem to offer fundamental insights, namely mathematics, physics, and philosophy.

Mathematics. We have argued elsewhere [Pra95a, Pra95b] that the mathematical universe is constructed from the interaction of polar opposites, specifically sets and "antisets" or Boolean algebras (of the complete and atomic kind when that question arises), in varying proportions.

Chu spaces are in a certain sense the dual of categories. This may be seen from the perspective of universal algebra, which organizes mathematics into three levels of abstraction, elements (numbers, points), structures (groups, topological spaces), and categories of structures (**Grp**, **Top**). This last may be organized as the (superlarge) category **ConCAT** of large *concrete* categories (C, U_C) where $U_C : C \to \mathbf{Set}$ is a faithful functor assigning a set or *carrier* to every object of the large category C.

Category theory reduces this organization to two levels by suppressing the elements of structures and calling the latter merely objects. Elements of objects x of C are recovered when needed as morphisms from $\top$ to x for a suitable object $\top$ of C, typically C's tensor unit.

Chu spaces also reduce the universal algebra picture to two levels, by suppressing not the elements but instead the boundaries between categories. This viewpoint applies to any category C that concretely and fully embeds a given class of categories; C contains all the objects of all categories in that class, and at least for pairs of objects of C realizing objects from the same category D,

the morphisms between those objects in C are exactly those between them in D, in the sense that they consist of the same underlying functions acting on the same underlying sets, by fullness and concreteness.

In addition to this mathematical universality, Chu spaces also have more specialized mathematical uses. For example they permit a more uniform treatment [Pra95b] of the gamut of categories treated by Johnstone under the heading of Stone spaces [Joh82]. Stone duality is poorly understood in computer science, yet is of crucial importance to understanding the relationships between imperative and declarative programming, denotational and operational semantics, and algebra and logic. By simplifying Stone duality to mere matrix transposition, Chu spaces make that subject more accessible.

Physics. The physical universe appears to be constructed from the interaction of particles and waves, an interaction described by quantum mechanics that makes everything a mixture of the two, again in varying proportions. Associating the points of Hilbert space with states and the dimensions (for a given choice of basis) with outcomes of measurements (events) associated to that basis as usual, a wavefunction as a pure state encodes correlations decoded with $\langle\psi|\varphi\rangle$ and $|\psi\rangle\langle\varphi|$, the notation in quantum mechanics for the respective residuals $\psi\backslash\varphi$ and ψ/φ. A mixed state, or mixture of wavefunctions described by a probability density matrix, may be understood as corresponding to a whole Chu space.

Philosophy. Yet another subject amenable to this perspective is the mind-body problem. Descartes proposed in 1637 that the mind interacted with the body. This proposal generated much literature all denying the causal interaction of mind and body and explaining their apparent interaction via various forms of *deus ex machina* (Malebranche, Spinoza, Leibniz), or denial of body (Berkeley) or mind (Hobbes), or the assertion of their equivalence (Russell). Elsewhere [Pra95a] we have applied Chu spaces to an implementation of Descartes' proposal, by taking the causal interaction of mind and body as basic instead of derived and obtaining as its consequences the basic interactions within each of body and mind. We do not see how to obtain the other direction, mind-body interaction from the separate behavior of mind and body, any better than did Descartes' contemporaries.

Viewed at the object level, Chu spaces formalize Russell's solution mathematically by offering dual views of the same Chu space merely by transposition of viewpoint. Viewed at the level of individual interactions within an object however, the solution takes on a new and deeper meaning: mind-body interaction turns out to be the *only* real interaction, body-body and mind-mind interaction are secondary interactions, derivable by residuation, that can be considered mere figments of our imagination as to how the universe interacts with itself.

6 Conclusion

Reflections. Our own interest in linear logic is as a process algebra. But linear logic was developed by a proof theorist, purportedly to tidy up proofs. Why

should such different goals lead to the same logic?

Linear logic is a resource sensitive logic in the sense that it disallows weakening (unused premises) and contraction (multiply used premises). Some have taken this to mean that it is a logic for reasoning *about* resources such as time and space in computation. It seems to us however that resource sensitivity in the logic is a dual side effect of resource *neglect* arising from reasoning inside the model, as results with equational logic and categorical reasoning. External reasoning in contrast takes place in the combinatorial world of sets, whose cartesian closed structure renders weakening and contraction sound, and where cardinality tracks resources [Pra95b, §7].

Resource sensitivity is merely a symptom, linear logic is more properly understood as a logic of interaction of polar opposites, with $\mathcal{A}\otimes\mathcal{B}$ as the interaction operator, $\mathcal{A}^\perp$ as the duality "mirror" interchanging opposites, and $!\mathcal{A}$ and $?\mathcal{A}$ as projecting the domain onto its respective poles. The remaining operator $\mathcal{A}\oplus\mathcal{B}$ and constant 0 then provide finite coproducts, with duality then yielding finite products (but why not coequalizers etc.?). We have argued [Pra95b] that **Set** and $\mathbf{Set}^{\mathrm{op}}$ suffice as poles in mathematical practice, but we acknowledge that there remains considerable room for debate here.

History. The Chu construction takes a symmetric monoidal closed category V with pullbacks and an object k of V and "completes" V to a self-dual category $\mathbf{Chu}(V,k)$. The details of the construction appear in P.-H. Chu's master thesis, published as the appendix to his advisor M. Barr's book introducing the notion of *-autonomous category [Bar79].

The intimate connection between linear logic and *-autonomous categories was first noticed by Seely [See89], furnishing Girard's linear logic [Gir87] with a natural constructive semantics. Barr then proposed the Chu construction as a source of constructive models of linear logic [Bar91].

The case $V = \mathbf{Set}$ is important for its combination of simplicity and generality. This case was first treated explicitly by Lafont and Streicher [LS91], where they treated its connections with von Neumann-Morgenstern games and linear logic, observing in passing that vector spaces, topological spaces, and coherent spaces were realizable as games, giving a small early hint of their universality.

Our own interest in Chu spaces was a consequence of attempts to formalize a suitable notion of partial distributive lattice. After arriving at such a notion based on the interaction of ordered Stone spaces and distributive lattices, we found that the resulting category was equivalent to both $\mathbf{Chu}(\mathbf{Set},2)$ and a full subcategory of $\mathbf{Chu}(\mathbf{Pos},2)$ for which the description and extension monotone functions of a Chu space were both full, i.e. isometries.

The name "Chu space" (over K) was suggested to the author by Barr in 1993 as a suitable name for the objects of $\mathbf{Chu}(\mathbf{Set},K)$ reifying "Chu construction," which predated Lafont and Streicher's "game." An advantage of "Chu space" is that it requires no disambiguating qualification to uniquely identify it, unlike "game." By analogy with categories enriched in V [Kel82] one might refer to the objects of the general Chu construction $\mathbf{Chu}(V,k)$ as *V-enriched* Chu spaces.

Acknowledgements

The application of Chu spaces to computation was developed in collaboration with my student Vineet Gupta [Gup94]. Rob van Glabbeek and Gordon Plotkin have separately been invaluable sounding-boards and sources of ideas, and jointly of results [VGP95]. Carolyn Brown, with Doug Gurr, was the first to explore the connection between the Chu construction and concurrency [BG90], and has been a useful source of insights. Dusko Pavlovic has developed a formal sense in which Chu spaces are universal for mathematics, more correctly their couniversality. I am also grateful to H. Andreka, I. Nemeti, I. Sain for the opportunity to present a one-week course on Chu spaces to an attentive audience at their TEMPUS summer school in Budapest in 1994. I am also very grateful to ONR for making it possible to host small two-monthly "Chu sessions" for the past three summers, involving our students along with Rob van Glabbeek and visitors Gordon Plotkin, Carolyn Brown, and this year Dusko Pavlovic and Michael Barr.

References

[Asp87] A. Asperti. A logic for concurrency. Manuscript, November 1987.

[Bar79] M. Barr. $*$-*Autonomous categories*, volume 752 of *Lecture Notes in Mathematics*. Springer-Verlag, 1979.

[Bar91] M. Barr. $*$-Autonomous categories and linear logic. *Math Structures in Comp. Sci.*, 1(2):159–178, 1991.

[BG90] C. Brown and D. Gurr. A categorical linear framework for Petri nets. In J. Mitchell, editor, *Logic in Computer Science*, pages 208–218. IEEE Computer Society, June 1990.

[BK84] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60:109–137, 1984.

[BK89] J.A. Bergstra and J.W. Klop. Process theory based on bisimulation semantics. In *Proc. REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, pages 50–122, Noordwijkerhout, The Netherlands, 1989. Springer-Verlag.

[CCMP91] R.T Casley, R.F. Crew, J. Meseguer, and V.R. Pratt. Temporal structures. *Math. Structures in Comp. Sci.*, 1(2):179–213, July 1991.

[GG89] C. A. Gunter and V. Gehlot. Nets as tensor theories. (preliminary report). In G. De Michelis, editor, *Applications of Petri Nets*, pages 174–191, 1989. Also University of Pennsylvania, Logic and Computation Report Number 17.

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[GN95] S. Gay and R. Nagarajan. A typed calculus of synchronous processes. In *Logic in Computer Science*, pages 210–220. IEEE Computer Society, June 1995.

[GP93] V. Gupta and V.R. Pratt. Gates accept concurrent behavior. In *Proc. 34th Ann. IEEE Symp. on Foundations of Comp. Sci.*, pages 62–71, November 1993.

[Gup93] V. Gupta. Concurrent Kripke structures. In *Proceedings of the North American Process Algebra Workshop, Cornell CS-TR-93-1369*, August 1993.

[Gup94] V. Gupta. *Chu Spaces: A Model of Concurrency*. PhD thesis, Stanford University, September 1994. Tech. Report, available as ftp://boole.stanford.edu/pub/gupthes.ps.Z.

[Hoa78] C.A.R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–672, August 1978.

[Joh82] P.T. Johnstone. *Stone Spaces*. Cambridge University Press, 1982.

[Kah74] G. Kahn. The semantics of a simple language for parallel programming. In *Proc. IFIP Congress 74*. North-Holland, Amsterdam, 1974.

[Kel82] G.M. Kelly. *Basic Concepts of Enriched Category Theory: London Math. Soc. Lecture Notes*. 64. Cambridge University Press, 1982.

[LS91] Y. Lafont and T. Streicher. Games semantics for linear logic. In *Proc. 6th Annual IEEE Symp. on Logic in Computer Science*, pages 43–49, Amsterdam, July 1991.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[Mil93] R. Milner. Action calculi, or syntactic action structures. In *Proceedings of MFCS'93*, volume 711 of *Lecture Notes in Computer Science*, pages 105–121, Gdańsk, Poland, 1993. Springer-Verlag.

[MPW92] R. Milner, J. Parrow, and D Walker. A calculus of mobile processes. *Information and Control*, 100:1–77, 1992.

[NPW81] M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures, and domains, part I. *Theoretical Computer Science*, 13:85–108, 1981.

[Pet62] C.A. Petri. Fundamentals of a theory of asynchronous information flow. In *Proc. IFIP Congress 62*, pages 386–390, Munich, 1962. North-Holland, Amsterdam.

[Pra85] V.R. Pratt. Some constructions for order-theoretic models of concurrency. In *Proc. Conf. on Logics of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 269–283, Brooklyn, 1985. Springer-Verlag.

[Pra86] V.R. Pratt. Modeling concurrency with partial orders. *Int. J. of Parallel Programming*, 15(1):33–71, February 1986.

[Pra92] V.R. Pratt. The duality of time and information. In *Proc. of CONCUR'92*, volume 630 of *Lecture Notes in Computer Science*, pages 237–253, Stonybrook, New York, August 1992. Springer-Verlag.

[Pra93] V.R. Pratt. The second calculus of binary relations. In *Proceedings of MFCS'93*, volume 711 of *Lecture Notes in Computer Science*, pages 142–155, Gdańsk, Poland, 1993. Springer-Verlag.

[Pra94a] V.R. Pratt. Chu spaces: Automata with quantum aspects. In *Proc. Workshop on Physics and Computation (PhysComp'94)*, Dallas, 1994. IEEE.

[Pra94b] V.R. Pratt. Time and information in sequential and concurrent computation. In *Proc. Theory and Practice of Parallel Programming (TPPP'94)*, Sendai, Japan, November 1994.

[Pra95a] V.R. Pratt. Rational mechanics and natural mathematics. In *TAPSOFT'95*, volume 915 of *Lecture Notes in Computer Science*, pages 108–122, Aarhus, Denmark, 1995. Springer-Verlag.

[Pra95b] V.R. Pratt. The Stone gamut: A coordinatization of mathematics. In *Logic in Computer Science*, pages 444–454. IEEE Computer Society, June 1995.

[Pri70] H.A. Priestley. Representation of distributive lattices. *Bull. London Math. Soc.*, 2:186–190, 1970.

[Ros90] K.I. Rosenthal. *Quantales and their applications*. Longman Scientific and Technical, 1990.

[Sco76] D. Scott. Data types as lattices. *SIAM Journal on Computing*, 5(3):522–587, 1976.

[See89] R.A.G Seely. Linear logic, $*$-autonomous categories and cofree algebras. In *Categories in Computer Science and Logic*, volume 92 of *Contemporary Mathematics*, pages 371–382, held June 1987, Boulder, Colorado, 1989.

[Sto36] M. Stone. The theory of representations for Boolean algebras. *Trans. Amer. Math. Soc.*, 40:37–111, 1936.

[Tra95] B. Trakhtenbrot. Origins and metamorphoses of the trinity: logic, nets, automata. In D. Kozen, editor, *Logic in Computer Science*, pages 506–507. IEEE Computer Society, June 1995.

[VGP95] R. Van Glabbeek and G. Plotkin. Configuration structures. In *Logic in Computer Science*, pages 199–209. IEEE Computer Society, June 1995.