# From Sequential Processes to Grid Computation

**Mark Burgin**
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095
Email: mburgin@math.ucla.edu

**Marc L. Smith**
Department of Computer Science
Colby College
Waterville, ME 04901-8858
Email: mlsmith@colby.edu

**Abstract:** *We introduce an extended model for view-centric reasoning, EVCR, that provides more comprehensive and flexible abstractions for representing actual concurrency. The theory of Communicating Sequential Processes (CSP) provides an interleaved semantics for reasoning about concurrency through a sequentialized trace of events recorded by an idealized observer. The theory of View-Centric Reasoning (VCR) extended CSP with notions of parallel events and views, and lazy observation, to represent the true concurrency of simultaneous events and multiple, possibly imperfect observers. But VCR could be more general still, since its events, like those of CSP, are instantaneous. This restriction precludes the possibility of observing events that partially overlap in time. EVCR permits partially overlapping events to be observed and recorded by a lazy observer. The result is a more general model of concurrency; one that is more appropriate for reasoning about network functioning and, in particular, grid computation.*

**Keywords:** concurrent computation, concurrent process, event, grid automaton, trace

## 1. Introduction

Hoare's Communicating Sequential Processes (CSP) is a model of computation that consists of two main components. CSP provides a process algebra for the specification of concurrent processes, and a process calculus for reasoning about properties of concurrent systems. The process calculus is based on a model of observation-based reasoning, using the metaphor of an observer recording a trace of observable events. The set of observable events link the process algebra and calculus, since they are the building blocks of both CSP process definitions and computational histories.

The motivation for view-centric reasoning (VCR) stemmed from a desire to preserve more information about the history of a computation by introducing lazy observation, and multiple, possibly imperfect observers. In particular, CSP is an interleaved model of concurrency, which means the observer records simultaneously events she observes in some sequentialized partial order, thus resulting in a decrease of entropy (i.e., imposing an order of events that did not actually exist during the computation) and this is not realistic. To overcome this situation, VCR traces were defined over multisets of observable events, rather than atomic events. The laziness stems from sparing the observer the stressful decision of what order to record simultaneous events. Furthermore, VCR distinguishes two types of trace: a computation's history, and its corresponding views. Since it is possible for a distributed computation to be observed by more than one observer, consequences of relativity theory provide for the possibility of different views of the same computation. Thus, the parallel event traces of VCR extend CSP in two important ways. First, VCR provides the entropy-preserving abstraction of parallel events, and second, it provides a model for associating a computation's history with it's multiple, possibly imperfect, but realistic views.

VCR's contribution to the theory of CSP is not as general as it could be; a model of true concurrency should support abstractions for events whose occurrence partially overlap in time. Thus, the possibility of observed event simultaneity is not merely a Boolean proposition, but rather a continuum: events A and B may overlap entirely, partially, or not at all. Extended VCR (EVCR), which is developed in this paper, is the next step in the evolution of CSP from a model that reduces concurrency to an interleaving of

sequential events, to a model of true concurrency that provides abstractions that represent degrees of interleaving. EVCR should prove especially useful for modeling and reasoning about properties of modern grid computation.

## 2. Systems for Concurrent Computations

Concurrent processes run in systems that have more than one device. The most advanced algorithmic model for such systems is a grid array and its theoretical counterpart – grid automaton. That is why we start with informal definitions of grid arrays and grid automata.

**Definition 1.** A *grid array* (GAR) is a system of information processing systems (computers, networks, imbedded systems, etc.), which are situated in a grid, called nodes; these nodes are optionally connected and interact with one another.

**Example 1.** The World Wide Web.

**Example 2.** A computer with all its devices as nodes: processor, RAM, keyboard, printer monitor, mouse, modem, etc.

**Example 3.** A natural neural network, such as the brain or the Central Nervous System (CNS) of a human being.

**Definition 2.** A *grid automaton* (GA) is a system of automata, which are situated in a grid, called nodes, are optionally connected and interact with one another.

**Example 4.** An artificial neural network.

**Example 5.** A cellular automaton.

**Definition 3.** A (*physical*) *distributed process* is a group of concurrent processes that go on in a GA (GAR).

Distributed processes represent dynamics of an active GAR.

**Definition 4.** An *active GA* is a GA in which events are occurring.

As grid automata are models of physical devices and systems, their functioning occurs in physical time. However, each GA has its system time defined by processes in it [1]. For each process (in a grid automaton) there are two types of time. External time is of some other process. Internal time is determined by the order in which events are happening. Time determines concurrency. However, in the next section, we build an abstract model of concurrency that depends on several temporal relations. This abstract model is correlated with time as a physical phenomenon or system characteristic elsewhere.

## 3. Concurrency Models: Local Aspect

At first, we consider concurrency of events that is basic for concurrent processes.

**Definition 5.** An *event* is an abstract entity that represents an element or a component of an abstract process.

**Definition 6.** An *embodied event* is a state transition of a GA node or edge.

**Definition 7.** An *observable embodied event* is a detectable state transition of a GA node or edge.

Not all events are observable if there are, for example, indistinguishable states. There are different relations between events in a process. The pivotal is the ordering relation. Analyzing real computational and other processes, we can distinguish several types of event pairs:

- *A sequential pair of events* consists of two events where one ends before the next starts.
- *A simultaneous pair of events* consists of two events where both events start and end simultaneously.
- *A coexisting pair of events* consists of two events where one starts before the next ends.
- *A separable pair of events* consists of two events that are not coexisting.
- *A separated pair of events* consists of two events for which coexistence is excluded.
- An event *r is included in* an event *q* if the event *q* starts before or when the event *r* starts and the event *q* ends after or when the event *r* ends.

Types of events are formally determined by the corresponding binary relations on sets of events:

- The *coexistence relation* $CE_R$ . This relation allows several interpretations, reflecting different modalities. It can formalize situations in which events must go so that one starts before another is finished (prescriptive coexistence). It can also formalize situations in which one event really starts before another is finished (actual coexistence). It can as well formalize situations in which one event may start before another is finished (possible coexistence).
- The *separability relation* $SP_R$ . This relation shows when events are (or may be) not coexisting.
- The *ordering relation* $OD_R$ . This relation allows several interpretations. It can formalize the order in which events must happen (prescriptive order), e.g., at first, we need to compute the value of a function, and only then to print the result. It can also formalize the order in which events really happened (actual order). It can as well formalize the order in which events may happen (possible order).
- The *simultaneity relation* $ST_R$ . This relation allows several interpretations. It can formalize situations in which events must go simultaneously (prescriptive coexistence). It can also formalize situations in which events really go simultaneously (actual coexistence). It can as well formalize situations in which events may go simultaneously (possible coexistence).
- The *inclusion relation* $IC_R$ . This relation allows several interpretations. It can formalize situations in which events must go so that the included event ends before or when the other is finished and starts after or when the other starts (prescriptive inclusiveness). It can also formalize situations in which the included event really ends before or when the other is finished and really starts after or when the other starts (actual inclusiveness). It can as well formalize situations in which it is possible that the included event ends before or when the other is finished and starts after or when the other starts (possible inclusiveness).

As EVCR actually models arbitrary processes, some illustrations of the above relations may be found in music for the piano, where the musical notes are events. A piece of music is composed from individual notes, each produced by a corresponding key on the piano, according to a musical prescription, represented in the form of sheet music. The sheet music prescribes when to play separated notes in succession (e.g., a run), and when to play notes in different possible coexisting fashions. Chords, arpeggios, melodies, harmonies, and syncopated rhythms are but a few examples of what can be prescribed by different combinations of the above relations. Upon performing such a piece of music, members of the audience subsequently make a determination of how well the actual composition of musical notes match what they believe was prescribed. In some cases, the result is a round of applause.

Returning our attention to grid automata, natural conditions on the above relations are derived from an analysis of real computations and other processes:

**(CP 1)** $CE_R$ is a complement of $SP_R$ .

**Corollary 1.** Any two events are either coexisting or separable.

**(CP 2)** $OD_R$ is a subset of $SP_R$ .

**Corollary 2.** Any two ordered events are separated.

**(CP 3)** $ST_R$ and $IC_R$ are subsets of $CE_R$ .

**Corollary 3.** Any two simultaneous events (of which one is included into another) are coexisting.

**(CP 4)** $CE_R$ is a tolerance, i.e., it is reflexive and symmetric.

**Corollary 4.** Any event is coexisting with itself.

**(CP 5)** $ST_R$ is an equivalence.

**Corollary 5.** Any event is simultaneous with itself.

**(CP 6)** $OD_R$ is a partial quasiorder.

**(CP 7)** $SP_R$ is reflexive and symmetric.

**(CP 8)** $IC_R$ is a tolerance.

**Corollary 6.** Any event is included into itself.

These relations define connections between events:

**Definition 8.** Two events *a* and *b* are *existentially connected* if $a CE_R^* b$.

**Proposition 1.** $CE_R^*$ is an equivalence relation.

**Definition 9.** Two events *a* and *b* are *sequentially connected* if $a OD_R b$.

**Definition 10.** Two events *a* and *b* are *parallelly connected* if $a ST_R b$.

Let us consider some important types of complex events.

**Definition 11.** A *parallel event* is a group of simultaneous events.

**Definition 12.** A *complete parallel event* is a group of all events that are simultaneous to one event from this group.

**Proposition 2.** A complete parallel event does not depend on the choice of the event to which all other events from this group are simultaneous.

**Proposition 3.** A complete parallel event is a maximal parallel event.

**Proposition 4.** If $ST_R = CE_R$ , then $OD_R$ induces an order relation on the set of all complete parallel events and it is possible to make this order linear.

**Definition 13.** A *coexisting event* is a group of events in which each pair of events is coexisting.

**Proposition 5.** Any parallel event is a coexisting event.

However, in general, not every coexisting event is a parallel event. These concepts coincide if and only if $ST_R = CE_R$ .

**Definition 14.** A *complete coexisting event* is a maximal coexisting event.

As the simultaneity relation $ST_R$ is a subset of the coexistence relation $CE_R$ , we have the following result.

**Proposition 6.** A complete coexisting event includes as subsets all parallel events of its elements.

**Proposition 7.** Two events *a* and *b* are simultaneous if and only if *a* is included in *b* and *b* is included in *a*, or formally, $b ST_R a \Leftrightarrow a IC_R b \;\&\; a IC_R b$.

**Corollary 7.** $ST_R = IC_R \cap IC_R^{-1}$.

## 4. Concurrency Models: Global Aspect

Events can be *elementary* and *complex*. Complex events consist of other events. This allows us to introduce hierarchy in the set of events and processes.

**Definition 15.** An *abstract process* is a system of related events.

Here events are related if for each two of them *q* and *p* it is possible to find a sequence of events $\{ r_1 , \dots , r_n \}$ such that $q = r_1$ , $p = r_n$ and each pair $(r_i , r_{i+1})$ belongs to, at least, one of the relations $IC_R$ , $ST_R$ , $OD_R$ , $SP_R$ , and $CE_R$.

**Definition 16.** An *embodied process* is a system of related embodied events.

Observation on a process gives a trace as its result.

**Definition 17.** A *trace tr* is a group of connected events.

Here we consider all three kinds of connections: existential, parallel, and sequential connection.

Let us consider some important types of traces.

**Definition 18.** A trace *tr* is called *sequential* if $OD_R$ is a linear order in it.

This concept coincides with the concept of trace in CSP [3].

**Definition 19.** A trace *tr* is called *parallel* if it is a parallel event.

**Definition 20.** A trace *tr* is called *sequentially parallel* if it consists of linear ordered parallel events.

This concept coincides with the concept of trace in VCR [6].

**Definition 21.** A *cluster* $Ct_{tr}(a)$ *of an event a* in a trace *tr* is a group of events coexisting with *a* in the trace *tr*. The event *a* is called the *base* of the cluster $Ct_{tr}(a)$.

By the definition, a cluster $Ct_{tr}(a)$ of an event *a* is a trace.

**Remark 1.** A cluster is not necessarily a coexisting event.

**Proposition 8.** A cluster is a coexisting event if and only if it does not contain two separable events.

**Definition 22.** A cluster $Ct_{tr}(a)$ of an event *a* is called *complete* if it contains all events coexisting with *a*. Complete cluster is denoted by $CCt_{tr}(a)$.

**Proposition 9.** Any coexisting event to which an event *a* belongs is a subset of the complete cluster $CCt_{tr}(a)$.

**Corollary 8.** Any parallel event to which an event *a* belongs is a subset of the complete cluster $CCt_{tr}(a)$.

**Definition 23.** A cluster $Ct_{tr}(a)$ of an event $a$ is called *ordered* if all events in $Ct_{tr}(a)\backslash\{a\}$ are ordered.

**Proposition 10.** In an ordered cluster $Ct_{tr}(a)$ of an event $a$, all but, at most, two events are included in $a$.

**Proposition 11.** If for events $a$ and $b$, $a\mathrm{IC_R}b$ is true, then $\mathrm{CCt}_{tr}(a) \subseteq \mathrm{CCt}_{tr}(b)$.

**Definition 24.**

a) An event $a$ in a trace $tr$ is called elementary (in $tr$) when any event coexisting with $a$ includes $a$.

b) An event $a$ in a trace $tr$ is called elementary for a set $E$ of events if any event from $E$ coexisting with $a$ includes $a$.

c) An event $a$ in a set $E$ of events is called elementary in $E$ when any event from $E$ coexisting with $a$ includes $a$.

**Proposition 12.** If $a\mathrm{IC_R}b$ and $a\mathrm{IC_R}c$, then $b\mathrm{CO_R}c$.

**Corollary 9.** If $a$ is an elementary event, $a\mathrm{CO_R}b$ and $a\mathrm{CO_R}c$, then $b\mathrm{CO_R}c$.

Proposition 7 implies the following result.

**Corollary 10.** Two coexisting elementary events are simultaneous.

**Proposition 13.** An event simultaneous with an elementary (for a set $E$) event is itself elementary (for $E$).

**Remark 2.** Usually only finite traces are considered. However, some models of computation include (and have to include to be adequate) infinite sets of events (cf., for example, [5, 7, 4, 2].

**Definition 25.** A (*elementary*) *section* of a trace $tr$ is a cluster $Ct_{tr}(a)$ in which the base $a$ is elementary (in the trace $tr$).

Corollary 9 implies the following result.

**Proposition 14.** Any section is a coexisting event.

**Definition 26.** A *parallel* (*elementary*) *section* of a trace $tr$ is a set of (elementary) sections with parallel bases.

It is possible to consider parallel section in the model VCR. There any parallel section consists of parallel events [6].

**Definition 27.** A *randomly ordered* parallel section, or ROPS, is a randomly ordered list of clusters from a parallel section.

It is possible to consider ROPS in the model VCR. There any ROPE coincides with a ROPS [6].

**Definition 28.** A *view* of a trace $tr$ is a list of ROPSs in the trace $tr$.

All introduced constructions of EVCR allow one to build VCR as submodel of EVCR. In this submodel only the simultaneity relation $\mathrm{ST_R}$ is considered because all events are instantaneous.

## 5. Conclusions and Future Work

We used the model of Grid Automata as the basis for extending View-Centric Reasoning, and introduced EVCR as the next step in VCR's evolution. The new definitions of events and the many ways events may coincide with other events form the basis for EVCR as a model that provides the abstractions needed for reasoning about properties of modern grid computation. In particular, EVCR provides for the

possibility of events A and B to overlap in varying degrees, instead of the all-or-nothing simultaneity of VCR's (and CSP's) instantaneous events. This added dimension of continuous time makes EVCR a more natural model for reasoning about Grid Automata, which must take time and location into account.

Still, we have only laid the foundation for a model that holds much promise. The next stages of EVCR will include the definition of composition of Grid Automata, including implications for composition of their respective parallel event traces. Longer term, EVCR may lead to the development of logical tools for reasoning about concurrent events and processes in EVCR.

Another perspective direction for the further development of EVCR is the automation of commonsense reasoning, a long goal of the field of artificial intelligence. EVCR provides a flexible base for enhancing the event calculus, which is used as an effective technique for commonsense reasoning [8,9].

## References


[1] M. Burgin, *Elements of the System Theory of Time*, LANL, Preprint in Physics 0207055, 2002, 21 p. (electronic edition: http://arXiv.org)

[2] M. Burgin, *Super-recursive Algorithms*, Springer, New York, 2005

[3] C. Hoare, *Communicating Sequential Processes*, Prentice Hall International Series in Computer Science. UK: Prentice-Hall International, UK, Ltd., 1985.

[4] Li, W., Ma, S., Sui, Y., and Xu, K. (2001) *A Logical Framework for Convergent Infinite Computations*, Preprint cs.LO/0105020 (electronic edition: http://arXiv.org)

[5] Rabin, M.O. (1969) Decidability of Second-order Theories and Automata on Infinite Trees, *Transactions of the AMS*, v. 141, pp. 1-35

[6] M. Smith. (2000) *View-Centric Reasoning about Parallel and Distributed Computation*. PhD thesis, University of Central Florida, Orlando, FL 32816-2362, December 2000.

[7] Vardi, M.Y. and Wolper, P. (1994) Reasoning about Infinite Computations, *Information and Computation*, v. 115, No.1, pp. 1—37

[8] Kowalski, R. and Sergot, M. J. (1986) A logic-based calculus of events, *New Generation Computing*, v. 4, pp. 67-95

[9] Mueller, Erik T. Commonsense reasoning, San Francisco, Morgan Kaufmann, 2006