Communicating Process Architectures – 2003
Jan F. Broenink and Gerald H. Hilderink (Eds.)
IOS Press, 2003

The Denotational Semantics of View-Centric Reasoning

Marc L. SMITH
Department of Computer Science, Colby College, Waterville, ME 04901-8858, USA
Charles E. HUGHES
School of EE and CS, University of Central Florida, Orlando, FL 32816-2362, USA
Kyle W. BURKE
Department of Computer Science, Colby College, Waterville, ME 04901-8858, USA

Abstract. Both Lawrence's $\mathcal{HCSP}$ [1] and Smith, et al's $\mathcal{VCR}$ [2] (an earlier version appears in [3]) extend CSP [4] with representations of truly concurrent events. Previously, $\mathcal{VCR}$ was described using an operational semantics, while the semantics of $\mathcal{HCSP}$'s Acceptances model, like those of the predominant CSP models described by Roscoe [5] (e.g., Traces, Failures / Divergences), are denotational. We now present a denotational semantics for $\mathcal{VCR}$ and, in so doing, propose an extension to $\mathcal{HCSP}$ (and possibly other existing CSP models) to support View-Centric Reasoning. This work brings $\mathcal{VCR}$ a step closer to being drawn within Hoare and He's Unifying Theories of Programming [6] for further comparisons.

1 Introduction

This paper represents a first step in the evolution of View-Centric Reasoning ($\mathcal{VCR}$) [2] from a model described by an operational semantics to one expressed in the more mainstream denotational semantics of other existing CSP models. In particular, Rosco [5] describes the modern Traces model ($\mathcal{T}$), the Failures/Divergences model ($\mathcal{N}$), the Stable Failures model ($\mathcal{F}$), the Infinite Traces/Divergences model ($\mathcal{I}$), and the Failures/Divergences/Infinite Traces model ($\mathcal{U}$) — all expressed in denotational semantics, and consistent with Hoare's classic CSP [4]. In addition, Lawrence recently proposed a new Acceptances model in two papers on $\mathcal{CSPP}$ [7] and $\mathcal{HCSP}$ [1], again, using denotational semantics.

The ultimate goal of transforming $\mathcal{VCR}$ into a denotational semantics is to provide a mechanism for drawing $\mathcal{VCR}$ into Hoare and He's Unifying Theories of Programming [6]. Accomplishing this task will provide a formal means of comparison between elements of $\mathcal{VCR}$ and those of the other CSP models. It so happens that $\mathcal{VCR}$ and $\mathcal{HCSP}$ share an important common abstraction: both models provide a means to directly represent simultaneously occurring events in their traces. In $\mathcal{HCSP}$, these are called *merged* events; in $\mathcal{VCR}$, they are called *parallel events*. For this reason, it was natural for us to propose $\mathcal{VCR}$ as an extension to $\mathcal{HCSP}$.

The remainder of this paper is organized as follows. Section 2 discusses the nature of environment in CSP (and other) models of concurrency, and the consequences for what it means to observe interactions between processes in such an environment. Section 3 presents a denotational representation of the $\mathcal{VCR}$ semantics we are proposing as an extension to $\mathcal{HCSP}$. We conclude and contemplate future work in Sections 4 and 5, respectively.

Attribute	Classic CSP	$\mathcal{HCSP}$	$\mathcal{VCR}$
Perfect observer?	Yes	Yes	Yes
Sequentially interleaved trace?	Yes		Yes
Trace with true simultaneous events?		Yes	Yes
Multiple (possibly) imperfect observers?			Yes
Multiple views of history?			Yes
Distinction between history and views?			Yes

Table 1: Taxonomy of attributes for three CSP models

2 Environmental and Observational Perspectives

Hoare and He present theories of reactive processes in their Unifying Theories of Programming [6]. The notion of environment is elucidated early in this presentation, as environment is essential to theories of reactive processes, examples of which include CSP and its derivative models. Essentially, the environment is the medium within which processes compute. Equivalently, the environment is the medium within which processes may be observed. The behavior of a sequential process may be sufficiently described by making observations only of its input/output behavior. In contrast, the behavior of a reactive process may require additional intermediate observations.

Regarding these observations, Hoare and He borrow insight from modern quantum physics. Namely, they view the act of observation to be an interaction between a process and one or more observers in the environment. Furthermore, the roles of observers in the environment may be (and often are) played by the processes themselves! As one would expect, an interaction between such processes often affects the behavior of the processes involved.

A process, in its role as observer, may sequentially record the interactions in which it participates. Recall participation includes the act of observation. Naturally, in an environment of multiple reactive processes, simultaneous interactions may be observed. Prior to $\mathcal{VCR}$ and $\mathcal{HCSP}$, recording conventions required simultaneous events to be recorded in some sequence, including random. Hoare and He thus define a *trace* as “the sequence of interactions recorded up to some given moment in time.”

In CSP, interactions take the form of communications between processes across channels. Table 1 gives a taxonomy of attributes across CSP, $\mathcal{HCSP}$, and $\mathcal{VCR}$. The table depicts the gradual departure from perfect observation and a sequentialized interleaving expression of concurrency, toward imperfect observation and a contextualized interleaving that more closely preserves true concurrency. In some sense, the attributes depicted in Table 1 provide a roadmap for the denotational semantics presented in Section 3.

Briefly, classic CSP has one perfect observer who records one trace of atomic events, possibly interleaved in cases of unsynchronized simultaneity. $\mathcal{HCSP}$ has one perfect observer who records one trace of merged events; obviating the need for, and eliminating the possibility of, sequential interleaving in the presence of an environment offering simultaneously occurring events. $\mathcal{VCR}$ has multiple, possibly imperfect observers. (It has been pointed out that with hiding and internal choice, classic CSP and $\mathcal{HCSP}$ can model imperfect observers, but this is true only in a restricted sense.) In $\mathcal{VCR}$, two kinds of traces are distinguished: a history and its corresponding views. The history type of trace is a sequence of unordered parallel events (event multisets). The view type of trace is a sequence of ordered parallel events (ROPEs) derived from a given computation’s history.

3 $\mathcal{VCR}$ Semantics

Our goal is to describe the views of a trace, but we do so gradually. Along the way we will address imperfect observation in Section 3.1, and differences in perspective for observing a simultaneously occurring event in Section 3.2. We shall see, in Section 3.3, that representing views of a trace requires further modification to $\mathcal{HCSP}$'s already extended notion of a trace. Finally, in Section 3.4, we relate traces and views to one another for further study.

3.1 Imperfect Observation

To account for imperfect observation, there needs to be a way to represent an observer missing (i.e., not observing) one or more atomic events that occur in parallel with other atomic events. Since both $\mathcal{VCR}$ and $\mathcal{HCSP}$ represent parallel events using multisets, we proceed from this construction.

Let B be a bag, that is, an event multiset. Thus, as a multiset,

$$B = \{\!|\ b_1, b_2, \ldots b_n\ |\!\}.$$

Borrowing the *merge* operator ($\diamond$) from $\mathcal{HCSP}$, we could equivalently represent b as a bag, thus

$$B = b_1 \diamond b_2 \diamond \cdots b_n.$$

We introduce the notion of the $pieces()$ of B as the powerset of B. In this case, the powerset would be the set of all *multiset* subsets of B. Thus, let

$$P = pieces(B) = \{p \mid p \subseteq B\}.$$

The elements of P represent the possibilities of imperfect observation. For all elements $p \in P$, we could enumerate the elements of p, in a fashion similar to the original B, thus

$$p = \{\!|\ p_1, p_2, \ldots p_k\ |\!\},$$

where $0 \leq k \leq n$. It is tempting to wish to represent the elements of P using the merge operator, but this possibility would seem to break down for those cases where p is either empty or a singleton multiset. Certainly, one can represent empty and singleton bags using multiset notation; the challenge is to find a meaningful representation for these two special cases of bags using the merge operator. One way to overcome this problem is to borrow from CSP's hiding operator and employ the invisible event, τ, as described by Roscoe [5]. Using τ in conjunction with $\diamond$, we can represent both empty and singleton bags. Thus, we can represent p as follows:

$$p = \begin{cases} \tau \diamond \tau & \text{if } p = \emptyset, \\ \tau \diamond p_1 & \text{if } |p| = 1, \text{ where } p_1 \in p, \\ p_1 \diamond p_2 \diamond \cdots p_k & \text{otherwise, where } \forall\, 1 \leq i \leq k \leq n. p_i \in p. \end{cases}$$

This use of τ is not entirely within the spirit of hiding internal events as defined by CSP, but the end result is it permits us to continue using the $\diamond$ operator as we continue to define the semantics of views. It should be noted for completeness that τ is an identity for $\diamond$, thus

$$\tau \diamond \tau \equiv \{\!|\ |\!\} \equiv \emptyset,$$

and

$$\tau \diamond p_i \equiv \{\!|\ p_i\ |\!\}.$$

3.2 Different Perspectives

In the previous section, neither representation of p, an element of the set of pieces of a bag or multiset, conveys any information about order. The $pieces()$ function merely accounts for the possibility that an observer need not be perfect. As discussed in Section 2, one of the tenets of $\mathcal{VCR}$ is that the environment of a concurrent computation consists of multiple observers. At a minimum, each communicating sequential process represents an observer of the computation in which that process participates.

In $\mathcal{VCR}$ and $\mathcal{HCSP}$, CSP's perfect observer is free to record the trace of a computation unencumbered by the burden, and consequences, of sequentially interleaving simultaneously occurring events. But the semantics of $\mathcal{VCR}$ more closely model the environment by using these parallel events to generate the many possible perspectives of observers within the environment. $\mathcal{VCR}$ thus has the notion of a ROPE, a randomly ordered parallel event.

Just as a bag B has many possible multisets of pieces, for a given set of pieces $p \in pieces(B)$, there are many possible orderings of the elements of p. The set of possible orderings (perspectives) for an observer of B can be defined using our definition of $pieces()$. Thus,

$$ropes(B) = \{\langle r_1, r_2, \ldots, r_k\rangle \mid r = \{\!|\, r_1, r_2, \ldots r_k \,|\!\} \in pieces(B) \wedge k = |r|\}.$$

The careful reader might be bothered by several points in this definition of $ropes()$. First, the set of orderings is a set of traces! But this is only natural, since an ordering implies some sort of list, and it so happens that a trace is nothing more than a list of observable events. This foreshadows the recursive nature of one possible definition of a view presented in Section 3.3.

The next points of concern with this definition of $ropes()$ have to do with whether the given definition really includes all possible orderings of all possible subsets of bag B? There are two levels of event generation — one explicit, the other implicit — defined by the $pieces()$ function. The explicit level determines the size of the subset, k, and specifically which k events are chosen from B. The implicit level concerns the unordered nature of multisets. For example, if $\{\!|\, a, b \,|\!\} \in pieces(B)$, then it is also true that $\{\!|\, b, a \,|\!\} \in pieces(B)$. Thus, every permutation of any trace found within $ropes(B)$ will also be an element of $ropes(B)$.

In keeping with the spirit of $\mathcal{HCSP}$'s $\diamond$ operator, we introduce an appropriately decorated *ordered merge* operator, $\vec{\diamond}$, which we use to define an alternative expression for $\mathcal{VCR}$'s ROPEs. Thus,

$$\langle r_1, r_2, \ldots, r_k\rangle \equiv r_1 \vec{\diamond}\, r_2 \vec{\diamond} \cdots r_k.$$

Now that we have given the definition for a ROPE, which is nothing more than a partially ordered bag or multiset, we may proceed to Section 3.3, where we construct a new kind of trace out of ROPEs.

3.3 Views of a Trace

CSP denotes a trace as "a sequence of symbols, separated by commas and enclosed in angular brackets." [4]. The meaning of this representation of a trace is that of a sequentialized, recorded history of the observable events of a computation. By introducing the $\diamond$ operator, $\mathcal{HCSP}$ extends the definition of a trace with the capability of recording merged events (bags), though it is common practice to omit the bag (multiset) notation for singletons and record the multiset as if it were an individual event. Similarly, $\mathcal{VCR}$ defines a trace as a list of parallel events. In both cases, this extended notion of a trace is that of a list of event multisets. From this common notion of a trace we can proceed to define a trace's views.

Suppose for some composition of processes, P, that trace $tr \in traces(P)$. Further suppose that tr is a sequence of bags, thus

$$tr = \langle b_1, b_2, \ldots b_n \rangle.$$

Then we could define $views(tr)$ as follows:

$$views(tr) = views(\langle b_1, b_2, \ldots b_n \rangle) = \{\langle r_1, r_2, \ldots r_n \rangle \mid r_i \in ropes(b_i) \, \forall \, 1 \leq i \leq n\}.$$

The definition of $views()$ falls out rather nicely at this point. This just specifies that the set of all views of a trace *tr* consists of views that are formed using a ROPE of each bag in *tr*.

If we dig a little deeper into the appearance of these views, there are two possible representations: a list of lists and a list of ordered bags. Both representations have benefits. The list of lists representation could be flattened, resulting in traces the original CSP observer could have recorded. Furthermore, the recursive nature of the list of lists form is elegant and appealing, obviating the need for a merge operator, and supporting a hierarchical, rather than flat, environment of observers. These authors believe this last point has implications for reasoning about composition. The list of ordered bags representation provides a convenient mapping to $\mathcal{HCSP}$, and warrants further study in the context of the unifying theories.

3.4 Views of All Traces

One of the tenets of $\mathcal{VCR}$ is the ability to distinguish a computation's history (trace) from its views, while relating instances of both notions to each other. To relate an instance of a computation's history to all its views, we introduce the set of TraceViews() of some concurrent process, P. Thus,

$$TraceViews(P) = \{\langle tr, vw \rangle \mid tr \in traces(P) \wedge vw \in views(tr)\}.$$

This says that $TraceViews()$ is a set of trace/view pairs. In particular, it is the set of all possible traces, and corresponding views of each possible trace, of some process P.

This is a very large set. It represents the cross product between every possible computation of process P, and every possible (including imperfect) view of every possible computation of P. From this set one could project just those elements that are the trace/view pairs of a single computation. Recall that knowing all possible views of a computation is not sufficient to unambiguously determine a computation's true history. For further discussion on this topic, see Smith et al. [2].

4 Conclusion

We presented elements of a denotational semantics for View-Centric Reasoning within the framework of $\mathcal{HCSP}$, while preserving a dual representation that links $\mathcal{VCR}$ to its original operational semantics. Both representations of views – as a list of ordered bags using the new $\overset{\rightarrow}{\diamond}$ operator, and as a list of lists (ROPEs) – should prove useful in different ways. The $TraceViews()$ set encapsulates the entire denotational semantics of $\mathcal{VCR}$, and is the starting point for attempting to draw $\mathcal{VCR}$ within the Unifying Theories of Programming. This will provide the necessary framework to compare $\mathcal{VCR}$ to the other CSP models.

5 Future Work

As a consequence of the analysis presented here, we are investigating whether the $TraceViews()$ set could be incorporated into the other CSP models. For example, the CSP traces model, $\mathcal{T}$, consists of a set of traces for a process, in addition to a set of axioms, closure properties, etc. Replacing the set of $traces()$ with $TraceViews()$ would require further revision, verification, and proofs. Similarly for the failures / divergences model, $\mathcal{N}$, the stable failures model, $\mathcal{F}$, the infinite traces / divergences model, $\mathcal{I}$, the failures / divergences / infinite traces model, $\mathcal{U}$, and the Acceptances model in $\mathcal{HCSP}$. The obvious model to attempt first would be $\mathcal{HCSP}$, since it already supports the notion of true parallel events.

6 Acknowledgements

Rebecca J. Parsons, of ThoughtWorks, Inc., Chicago, was our collaborator during the development of the $\mathcal{VCR}$ model. Her guidance and patience helped lead to $\mathcal{VCR}$'s original expression as a parameterized operational semantics. Professor Adrian Lawrence, of Loughborough University, helped with terminology and clarification of some of the finer points of merged events in $\mathcal{HCSP}$. Professor Jim Woodcock, of the University of Kent, provided valuable feedback for our paper last year [3], as well the roadmap we are now following for $\mathcal{VCR}$. Finally, the authors wish to thank the anonymous referees of this paper for their careful reading and insightful remarks.

References

[1] Adrian E. Lawrence. Hcsp: Imperative state and true concurrency. In J. S. Pascoe, P. H. Welch, R. J. Loader, and V. S. Sunderam, editors, *Communicating Process Architectures – 2002*, Concurrent Systems Engineering, pages 39–55, Amsterdam, 2002. IOS Press.

[2] Marc L. Smith, Rebecca J. Parsons, and Charles E. Hughes. View-centric reasoning for linda and tuple space computation. *IEE Proceedings–Software*, 150(2):71–84, apr 2003.

[3] Marc L. Smith, Rebecca J. Parsons, and Charles E. Hughes. View-centric reasoning for linda and tuple space computation. In J. S. Pascoe, P. H. Welch, R. J. Loader, and V. S. Sunderam, editors, *Communicating Process Architectures 2002*, volume 60 of *Concurrent Systems Engineering Series*, pages 223–254, Amsterdam, 2002. IOS Press.

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

[5] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall International Series in Computer Science. Prentice Hall Europe, 1998.

[6] C.A.R. Hoare and Jifeng He. *Unifying Theories of Programming*. Prentice Hall Series in Computer Science. Prentice Hall Europe, 1998.

[7] Adrian E. Lawrence. Acceptances, behaviours, and infinite activity in cspp. In J. S. Pascoe, P. H. Welch, R. J. Loader, and V. S. Sunderam, editors, *Communicating Process Architectures – 2002*, Concurrent Systems Engineering, pages 17–38, Amsterdam, 2002. IOS Press.