1 The Constructive Method

1. A Model of Computation
2. Logic to encode propositions
3. Evidence Semantics

As an example involving languages and parsers:

1. For the proposition w:word $\in$ [G:Grammar] the evidence is a *parse tree*.
2. For the proposition $\forall w$:word $w \in [G] \vee w \notin [G]w$ the evidence is a *parser*.
3. For the proposition $\forall G : LALR(1) \forall w$:word $w \in [G] \vee w \notin [G]$ the evidence is a *parser generator*.

2 A Logic For Distributed Systems

Use the logic of Message Sequence Diagrams (MSDs) already used by programmers to reason about real-world distributed systems. A MSD can be considered an Event Ordering of Events (E), assignments of locations (L) to events and Lamport's Causal Ordering Relation (α): $\langle E, L : E \rightarrow L, \alpha \rangle$

2.1 Axioms

1. $=, <$ (equality and before) are decidable
2. $<$ is transitive, strongly well founded and locally finite
3. $<$ is a total ordering of events at the same location

3 A Model of Computation for Distributed Systems

The model is that of Communicating Processes based on Processes, Messages, and Locations which can be combined to give Components and Systems, all executing in an Environment. Critically, Messages may contain other Processes or some information that can be used to extract a Process. We wish to give definitions of these concepts in terms of standard λ calculus and type theory so that existing machinery (like NuPRL) can be applied to reason about them.

In the following definitions, a *Bag* is an unordered collection of items allowing duplicates (hence neither a list nor a set).

3.1 Model Definitions

1. Process $\subseteq$ Msg $\rightarrow$ Process $\times$ Bag (Loc $\times$ Msg)
2. Component = Location $\times$ Process
3. System = Bag Component $\times$ Bag (Loca $\times$ Msg)
4. An Environment is responsible for deciding which message is to be delivered next

3.2 Co-Recursive Types

Co-Recursive Types are required in order to encode the property of extracting Processes from the contents of messages.

If F is a function from types to types (F::Type $\rightarrow$ Type) and $\top$ is the universal type, then

$$corec(T.F[T]) = \bigcap_{n:\mathbb{N}} F^n(\top)$$

For example, $corec(T.\mathbb{Z} \times T)$ is the type of streams of integers.

F is continuous if

$$\bigcap_{n:\mathbb{N}} FX_i \subseteq F(\bigcap_{n:\mathbb{N}} X_i)$$

F is monotone if

$$A \subseteq B \rightarrow F(A) \subseteq F(B)$$

3.3 Model Definitions using Types

Using the notion of corecursive types we can restate the above definitions. Assume that P is some data in a message that can be used to construct a process. Parentheses are used in definitions and square brackets in applications.

1. Process $\subseteq$ Msg(Process) $\rightarrow$ Process $\times$ Bag(Loc$\times$Msg(Process))
2. M(P) = Header $\times$ (Data + P)
3. Ext(P) = Bag (Loc $\times$ M[P])
4. F(P) = M[P] $\rightarrow$ P $\times$ Ext[P]
5. Process = corec(P.F[P])
6. Msg = M(Process)
7. Environment = $\mathbb{N} \rightarrow b$: Bag(Loc $\times$ Msg) $\rightarrow b_1, b_2$: Ext $\times$ Ext such that
   (a) $b_1 + b_2 = b$
   (b) There is no repeat in the locations of b_1

This then provides the basic required to encode configurations of the system as groups of processes and generate the evidence semantics which can in turn be used to reason about the behavior of such systems.