Constraint Nets: A Semantic Model for Hybrid Dynamic Systems

Ying Zhang and Alan K. Mackworth*

Department of Computer Science
University of British Columbia
Vancouver, B.C., Canada V6T 1Z4
E-mail: zhang, mack@cs.ubc.ca

Abstract

Hybrid dynamic systems are systems consisting of a non-trivial mixture of discrete and continuous components, such as a controller realized by a combination of digital and analog circuits, a robot composed of a digital controller and a physical plant, or a robotic system consisting of a computer-controlled robot coupled to a continuous environment. Hybrid dynamic systems are more general than traditional real-time systems. The former can be composed of continuous subsystems in addition to discrete and event-controlled components.

In this paper, we develop a semantic model, constraint nets (CN), for hybrid systems. CN captures the most general structure of dynamic systems so that systems with discrete and continuous time, discrete and continuous variables, and asynchronous as well as synchronous event structures, can be modeled in a unitary framework. Using aggregation operators, a system can be modeled hierarchically in CN; therefore, the dynamics of the environment as well as the dynamics of the plant and the dynamics of the controller can be modeled individually and then integrated. Based on abstract algebra and topology, CN supports multiple levels of abstraction, so that a system can be analyzed at different levels of detail. CN also provides a rigorous formal programming semantics for the design of hybrid real-time embedded systems.

1 Motivation and Introduction

A dynamic system is defined on a structure $\langle \mathcal{T}, A \rangle$ where $\mathcal{T}$ is a time structure and A is a domain structure; the time and domain structures can be either continuous or discrete. Table 1 shows examples of four basic types of models of dynamic systems.

We call a dynamic system composed of components of more than one basic type a *hybrid system*, for example, a controller realized by a combination of digital and analog circuits, a robot composed of a digital controller and a physical plant, and a robotic system consisting of a computer-controlled robot coupled to a continuous environment. Hybrid dynamic systems are more general than traditional

*Shell Canada Fellow, Canadian Institute for Advanced Research

Table 1: Examples of the basic types of models of dynamic systems

Dynamic Systems	*Discrete Time*	*Continuous Time*
Discrete Domain	Finite State Machines	Asynchronous Circuits
Continuous Domain	Difference Equations	Differential Equations

real-time systems. The former can be composed of continuous subsystems in addition to discrete and event-controlled components.

The development of models for hybrid systems has been very active over the last two years [20, 15, 21, 3]. We take a different approach to the study of hybrid systems. Our approach is motivated by the following arguments. First, hybrid systems consist of interacting discrete and continuous components. Instead of fixing a model with particular time and domain structures, a model for hybrid systems should be developed on both abstract time structures and abstract data types. Second, hybrid systems are complex systems with multiple components. A model for hybrid systems should support hierarchy and modularity. Third, hybrid systems are generalizations of basic discrete or continuous systems. A model for hybrid systems should be at least as powerful as existing computational models. In short, the model should be unitary, modular, and powerful.

We start with a general definition of time as a linearly ordered set with an initial time point, a metric space and a measure space. Then we examine domain structures in abstract algebra and topology. With any time structure and domain structure, we can define basic types of elements in dynamic systems: traces, which are functions of time, and transductions, which are mappings from traces to traces. The constraint net model (CN) is then developed on an abstract dynamics structure composed of a trace space and a set of basic transductions: transliterations, which are memoryless combinational processes, unit delays and transport delays, which are for sequential processes, and event-driven transductions. Event-driven transductions play an important role in this model as channels between continuous and discrete time components, or as synchronizers among asynchronous components.

The syntax of a constraint net is a bipartite graph with two types of nodes: locations and transductions, and a set of connections between locations and transductions. A constraint net can be composed hierarchically via modular and aggregation operators. Semantically, a constraint net represents a set of equations, with locations as variables and transductions as functions. The semantics of the constraint net, with each location denoting a trace, is the least fixpoint of the set of equations. The semantics of a system can be obtained hierarchically from the semantics of its components and internal connections. In this model, temporal integration is defined on vector spaces using infinitesimal transport delays.

CN is a deterministic dynamic process model; nondeterminism can be modeled via hidden or uncontrolled inputs. Thus, while more powerful, and simpler, than most inherently nondeterministic models, probabilistic and stochastic analysis can be incorporated. CN is also an abstract and general dynamic process model, while discrete state machines and differential state equations are particular instantiations of the model.

In summary, CN satisfies our objective which is to provide a model that is formal and general, modular and composite, as well as powerful and practical.

The rest of the paper is organized as follows. Section 2 introduces the syntactic structure of constraint nets. Section 3 develops the topological structure of dynamic systems. Section 4 presents the semantics of constraint nets using fixpoint theory and defines temporal integration using infinitesimal transport delays. Section 5 discusses modeling in constraint nets. Section 6 surveys the existing hybrid system models. Section 7 concludes this paper and points out related research. Appendix A presents the mathematical preliminaries; the proofs of theorems and propositions are in Appendix B.

2 Syntactic Structure of Constraint Nets

In this section, we introduce the syntax of constraint nets and characterize the composite structure and modularity of the model.

2.1 Syntax

Intuitively, a constraint net consists of a finite set of locations, a finite set of transductions and a finite set of connections. Each location is of fixed sort; a location's value typically changes over time. A location can be regarded as a wire, a channel, a variable, or a memory location. Each transduction is a causal mapping from inputs to outputs over time, operating according to a certain reference time or activated by external events. Connections relate locations with ports of transductions. A clock is a special kind of location which connects to the event port of an event-driven transduction.

Syntactically, a *constraint net* is a triple $CN = \langle Lc, Td, Cn \rangle$, where Lc is a finite set of *locations*, each of which is associated with a sort; Td is a finite set of labels of *transductions*, each of which is associated with a set of *input ports* and an *output port* and each port is associated with a certain sort; Cn is a set of *connections* between locations and ports of transductions of the same sort, with the following restrictions: (1) there is at most one output port connecting to each location, (2) each port of a transduction connects to a unique location and (3) no location is isolated.

A location l is an *output location* of a transduction F iff there is a connection between the output port of F and l; l is an *input location* of F iff there is a connection between an input port of F and l.

A location is an *output* of the constraint net if it is an output location of a transduction otherwise it is an *input*. The set of input locations of a constraint net CN is denoted by $I(CN)$, the set of output locations is denoted by $O(CN)$. A constraint net is *open* if there is an input location otherwise it is *closed*.

A constraint net is represented by a bipartite graph where locations are depicted by circles, transductions by boxes and connections by arcs. For example, Figure 1, where f is a transliteration of a state transition function and δ is a unit delay, is an open net, which can represent a state automaton: $s(0) = s_0, s(n+1) = f(i(n), s(n))$, given time as the set of natural numbers. Figure 2 is a closed net,

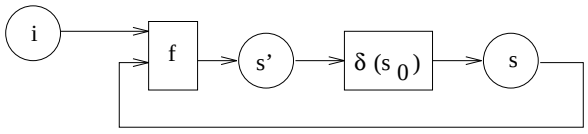

Figure 1: The constraint net representing a state automaton

which can represent a differential equation $\dot{s} = f(s)$, given time as a left-closed real interval.

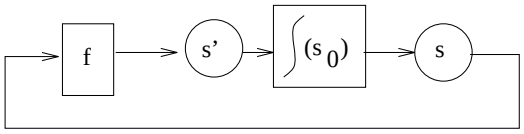

Figure 2: The constraint net representing $\dot{s} = f(s)$

2.2 Modules

A *module* is a triple $\langle CN, I, O \rangle$, denoted by $CN(I, O)$, where CN is a constraint net, $I \subseteq I(CN)$ and $O \subseteq O(CN)$ are subsets of the input and output locations of CN, respectively; $I \cup O$ defines the *interface* of the module. A module $CN(I, O)$ is *closed* if $I = \emptyset$ otherwise it is *open*. Locations in $I(CN) - I$ are *hidden inputs* and locations in $O(CN) - O$ are *hidden outputs*. A module is *nondeterministic* iff $I \subset I(CN)$. Graphically, a module is represented by a box with rounded corners.

A compound module can be constructed from simple ones. There are three basic operations that can be applied to modules to obtain a new module. The first is *union*, which generates a new module from two modules, with these two modules side by side. The second is *coalescence*, which coalesces two locations in the interface of a module into one location, with the restriction that at most one

of these two locations is an output location. The third is *hiding*, which deletes a location from the interface.

In addition to the three basic operations, we can define three combined operations. The first is *cascade connection* which connects two modules in series. The second is *parallel connection* which connects two modules in parallel. The third is *feedback connection* which connects an output of the module to an input of its own (Figure 3).

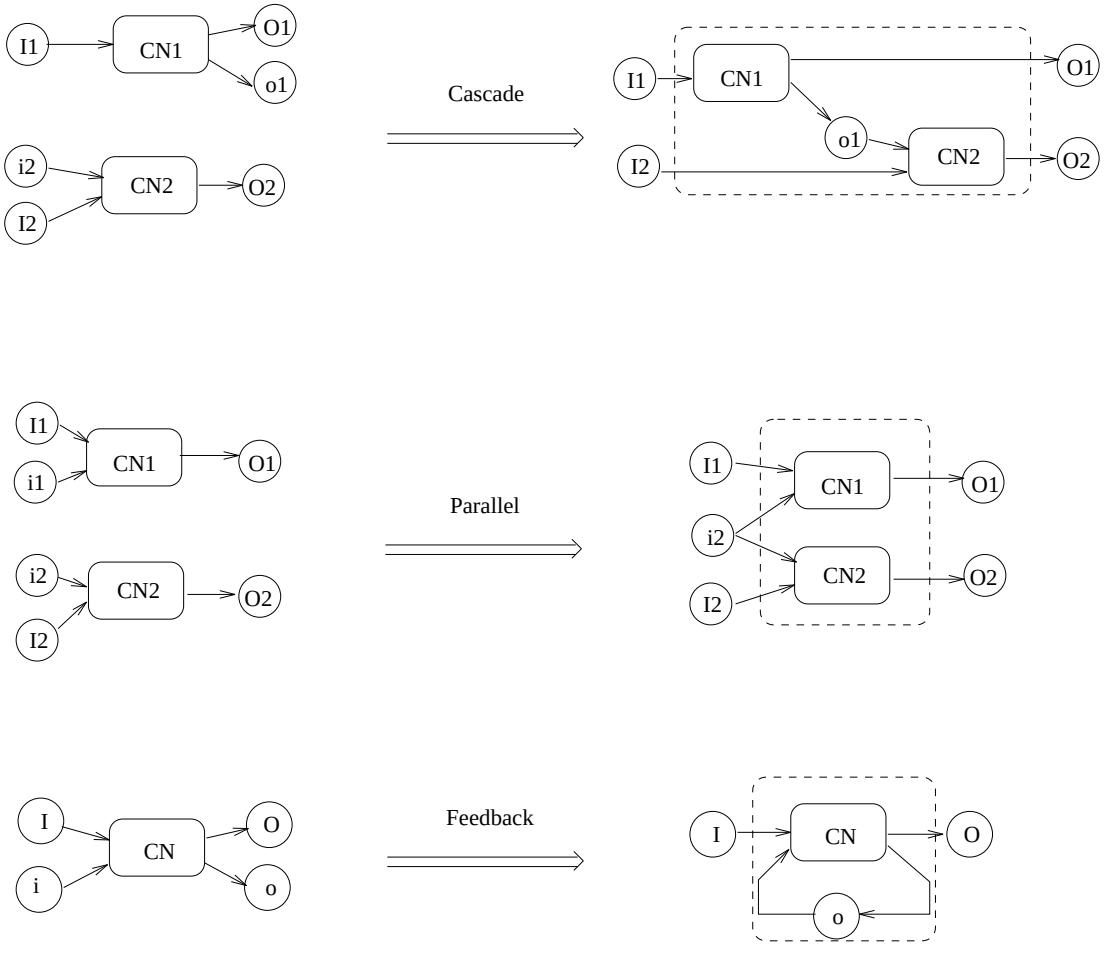

Figure 3: Cascade, parallel and feedback connections

There are three reasons for introducing modules.

First, modules create a hierarchical composition structure for complex systems. For example, a state automaton module SA is created if we select locations i, s or i, s' of the constraint net in Figure 1 as the interface. An input/output automaton IOA is then developed using a cascade connection of

SA and a transliteration g (Figure 4). IOA defines a transduction from input traces to output traces.

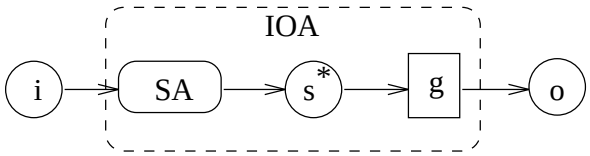

Figure 4: An input/output automaton (s^* denotes either s or s')

Second, modules provide a flexible way for generating different systems from the same set of components. To illustrate this idea, we again look at input/output automata. There are two types of input/output automata: Mealy machines [17] or Moore machines [19]. In general, an input/output automaton is a tuple $\langle \mathcal{I}, \mathcal{S}, s_0, f_s, \mathcal{O}, f_o \rangle$ where $\mathcal{I}$ is the set of input values, $\mathcal{S}$ is the set of states with $s_0 \in \mathcal{S}$ as the initial state, $f_s : \mathcal{I} \times \mathcal{S} \rightarrow \mathcal{S}$ is a state transition function, $\mathcal{O}$ is the set of output values and f_o is an output function. However, there are two ways to define an output function: $f_o : \mathcal{I} \times \mathcal{S} \rightarrow \mathcal{O}$ for Mealy machines and $f_o : \mathcal{S} \rightarrow \mathcal{O}$ for Moore machines. For a constraint net model of an input/output automaton, a Mealy or Moore machine is deduced by selecting different output locations as the interface of its state automaton module. If s' is selected, IOA is a Mealy machine with $f_s = f$ and $f_o = g \circ f$; if s is selected, IOA is a Moore machine with $f_s = f$ and $f_o = g$.

Third, modules capture the notion of abstraction via hidden locations. Clearly, hidden outputs encapsulate internal states of a system. However, the role of hidden inputs is not obvious. Consider again the state automaton in Figure 1. If the only input location i is chosen to be hidden, a module of a closed nondeterministic state transition system is generated. In particular, the state transition function f defines a state transition relation $R \subseteq \mathcal{S} \times \mathcal{S}$, $(s, s') \in R$ iff $\exists i \in \mathcal{I}, s' = f(i, s)$, or equally, the set of next possible states of a state s is $\{f(i, s) | i \in \mathcal{I}\}$. In general, any module $CN(I, O)$ with $I \subset I(CN)$ defines a nondeterministic system. Thus, while more powerful, and simpler, than most inherently nondeterministic models, probabilistic and stochastic analysis can be incorporated. Similar concepts have been explored earlier in general systems theory [18].

3 Topological Structure of Dynamic Systems

In this section, we discuss the topological structure of time, domains, traces and transductions, which are basic elements of dynamic systems. The mathematical preliminaries on general topology, partial orders and metric spaces are presented in Appendix A.

3.1 Time structures

Understanding time is the key to understanding dynamics. We formalize time using abstract structures which capture the important aspects of time: linearity, metric and measure. A time structure, in general, can be considered as a linearly ordered set with a start time point, an associated metric for quantifying the distance between any two time points and a measure for estimating the duration of time. Formally, a *time structure* is a triple $\langle \mathcal{T}, d, \mu \rangle$, where

- $\mathcal{T}$ is a linearly ordered set $\langle \mathcal{T}, \leq \rangle$ with $\mathbf{0}$ as the least element,
- $\langle \mathcal{T}, d \rangle$ forms a metric space with d as a metric satisfying: $\forall t_0 \leq t_1 \leq t_2$,

$$d(t_0, t_2) = d(t_0, t_1) + d(t_1, t_2),$$

- $\langle \mathcal{T}, d, \mu \rangle$ forms a measure space with μ as a Borel measure.

Let $[t_1, t_2) = \{t | t_1 \leq t < t_2\}$. Clearly μ is defined for all sets with form $[\mathbf{0}, t)$ and for the total set $\mathcal{T}$. Furthermore, we let $\mu([t_1, t_2)) = \mu([\mathbf{0}, t_2)) - \mu([\mathbf{0}, t_1))$ and $m(t) = d(\mathbf{0}, t)$. For simplicity, we will use $\mathcal{T}$ to refer to time structure $\langle \mathcal{T}, d, \mu \rangle$ when no ambiguity arises.

A time structure $\mathcal{T}$ is *discrete* iff $\mathcal{T}$ has no Cauchy sequence. $\mathcal{T}$ is *dense* iff for all $t_1 < t_2$, there exists t_0, such that $t_1 < t_0 < t_2$; it is *continuous* iff its metric space is connected. Clearly, a continuous time is also dense, but not *vice versa*. For example, $\mathcal{N}$, the set of natural numbers, and $\mathcal{R}^+$, the set of nonnegative real numbers, with $d(t_1, t_2) = |t_1 - t_2|$ and $\mu([0, t)) = t$, are time structures. $\mathcal{N}$ is discrete and $\mathcal{R}^+$ is continuous. The set of rational numbers $\mathcal{Q}$ with the same metric and measure forms a dense time structure. The set $\{\frac{2^n - 1}{2^n} | n \in \mathcal{N}\} \cup \{1\}$ or $\{0\} \cup \{\frac{1}{2^n} | n \in \mathcal{N}\}$ with the same metric and measure defines a time structure which is neither discrete nor dense. Even though our definition of time structures is extremely general, discrete or continuous time structures are most commonly used.

A time structure $\mathcal{T}$ may be related to another time structure $\mathcal{T}_r$ by a *reference time mapping* $h : \mathcal{T} \to \mathcal{T}_r$ where

- the order among time points is preserved: $t < t'$ implies $h(t) <_r h(t')$,
- the least element is preserved: $h(\mathbf{0}) = \mathbf{0}_r$,
- the distance between two time points is preserved: $d(t_1, t_2) = d_r(h(t_1), h(t_2))$, and
- the measure on any finite interval of time points is preserved: $\mu([\mathbf{0}, t)) = \mu_r([\mathbf{0}_r, h(t)))$.

$\mathcal{T}_r$ is a *reference time* of $\mathcal{T}$, and $\mathcal{T}$ is a *sample time* of $\mathcal{T}_r$. For example, if $h : \mathcal{N} \to \mathcal{R}^+$ is defined as $h(n) = n$, $\mathcal{R}^+$ is a reference time of $\mathcal{N}$. For any time structure $\mathcal{T}$, a reference time of $\mathcal{T}$ is as "dense" as $\mathcal{T}$.

3.2 Domain structures

As with time, we formalize domains as abstract structures so that discrete as well as continuous domains are defined uniformly. A domain can be a simple domain or a composite domain.

A *simple domain* is a pair $\langle A \cup \{\perp_A\}, d_A\rangle$ where A is a set, $\perp_A \notin A$ means undefined, and d_A is a metric on A. Let $\overline{A} = A \cup \{\perp_A\}$. For simplicity, we will use $\overline{A}$ to refer to simple domain $\langle \overline{A}, d_A\rangle$ when no ambiguity arises. For example, let $\mathcal{R}$ be the set of real numbers, $\overline{\mathcal{R}}$ is a simple domain with connected metric space; let $\mathcal{B} = \{0, 1\}$, $\overline{\mathcal{B}}$ is a simple domain with discrete topology on $\mathcal{B}$.

Any simple domain $\overline{A}$ is associated with a partial order relation $\leq_{\overline{A}}$. $\langle \overline{A}, \leq_{\overline{A}}\rangle$ is a flat partial order with $\perp_A$ as the least element, i.e., if $a_1 \leq_{\overline{A}} a_2$, then either $a_1 = a_2$ or $a_1 = \perp_A$. In addition, $\overline{A}$ is associated with a *derived metric topology* τ. Let the metric topology on A induced by the metric d_A be τ_A, τ is $\tau_A \cup \{\overline{A}\}$. A simple domain can also be represented as a triple $\langle \overline{A}, \leq_{\overline{A}}, \tau\rangle$ where $\leq_{\overline{A}}$ is the partial order relation and τ is the derived metric topology.

A domain is defined recursively based on simple domains. $\langle A, \leq_A, \tau\rangle$, with $\leq_A$ as the partial order relation and τ as the derived metric topology, is a *domain* iff:

- it is a simple domain; or
- it is a *composite* domain, i.e. it is the product of a family of domains $\{\langle A_i, \leq_{A_i}, \tau_i\rangle\}_{i\in I}$ such that $\langle A, \leq_A\rangle$ is the product partial order of the family of partial orders $\{\langle A_i, \leq_{A_i}\rangle\}_{i\in I}$ and $\langle A, \tau\rangle$ is the product space of the family of spaces $\{\langle A_i, \tau_i\rangle\}_{i\in I}$.

Note that we have no restriction on the index set I which can be arbitrary (finite or infinite, countable or uncountable etc.). For simplicity, we will use A to refer to domain $\langle A, \leq_A, \tau\rangle$ when no ambiguity arises. For example, let $n \in \mathcal{N}$ be some natural number, $\overline{\mathcal{R}}^n$ is a composite domain with n components; $\mathcal{N} \to \overline{\mathcal{B}}$, or equally, $\overline{\mathcal{B}}^{\mathcal{N}}$, is a composite domain with infinitely many components.

We take a signature as a syntactical structure of a class of multi-sorted domains with associated functions defined on these domains. Let $\Sigma = \langle S, F\rangle$ be a *signature* where S is a set of *sorts* and F is a set of *function symbols*. F is equipped with a mapping *type*: $F \to S^* \times S$ where S^* denotes the set of all finite tuples of S. For any $f \in F$, $type(f)$ is the type of f. We use $f : s^* \to s$ to denote $f \in F$ with $type(f) = \langle s^*, s\rangle$. For example, the signature of Boolean algebra can be described as: $\Sigma_b = \langle b, \{\mathtt{0}, \neg, \wedge, \vee\}\rangle$ with $\mathtt{0} :\to b$, $\neg : b \to b$, $\wedge : b, b \to b$, and $\vee : b, b \to b$. Σ_b has one sort with a constant $\mathtt{0}$ (nullary function), a unary function $\neg$, and two binary functions $\wedge$ and $\vee$.

A domain structure of some signature is defined as follows. Let $\Sigma = \langle S, F\rangle$ be a signature. A Σ-*domain structure* A is a pair $\langle \{A_s\}_{s\in S}, \{f^A\}_{f\in F}\rangle$ where for each $s \in S$, A_s is a domain of sort s, and for each $f : s^* \to s \in F$ with $s^* : I \to S$ and $s \in S$, $f^A : \times_I A_{s_i^*} \to A_s$ is a function denoted by

f. For example, $\langle \overline{\mathcal{B}}, \{0, \neg, \wedge, \vee\}\rangle$ is a Σ_b-domain structure where $\neg$, $\wedge$ and $\vee$ are negation, conjunction and disjunction, respectively.

3.3 Trace and event structures

A *trace* $v : \mathcal{T} \to A$ is a mapping from time $\mathcal{T}$ to domain A. For example if $\mathcal{T} = \mathcal{R}^+$ and $A = \overline{\mathcal{R}}$, $v_1 = \lambda t.\sin(t)$ and $v_2 = \lambda t.e^{-t}$ are traces.

A trace provides complete information at every finite point of time. Values at infinite time points are not represented explicitly, however, they can be derived when limits are introduced. For example, $\lim_{t\to\infty} \sin(t) = \perp_{\mathcal{R}}$ and $\lim_{t\to\infty} e^{-t} = 0$.

Given any linear order L and a domain A, $v : L \to A$ is a *linear set of values*. A value $v^* \in A$ is a *limit* of a linear set of values v, written $v \to v^*$, iff v^* is a limit of v in the derived metric topology. With this definition, the set of limits of v has the following properties.

Proposition 3.1 *If $v : L \to \overline{A}$, then*

(1) $v \to \perp_A$, and

(2) $v \to v_1^*$ and $v \to v_2^*$ imply that either $v_1^* \leq_{\overline{A}} v_2^*$ or $v_2^* \leq_{\overline{A}} v_1^*$.

Proposition 3.2 *If $v : L \to A$ for $A = \times_I A_i$, then*

(1) $v \to v^*$ iff $v_i \to v_i^*$ for all $i \in I$, and

(2) the set of limits $\{v^* | v \to v^*\}$ is a directed subset in $\langle A, \leq_A\rangle$ and has a least upper bound.

Therefore, we can define the limit of a linear set of values as follows. Let $v : L \to A$, the *limit* of v, written $\lim v$, is defined as the least upper bound of the set of limits of v, i.e. $\lim v = \bigvee_A \{v^* | v \to v^*\}$. The limits have the following properties.

Proposition 3.3 *If $v : L \to A$ for $A = \times_I A_i$, then $(\lim v)_i = \lim v_i, \forall i \in I$.*

Proposition 3.4 *If $v_1, v_2 : L \to A$ and $v_1(l) \leq_A v_2(l)$ for all $l \in L$, then $\lim v_1 \leq_A \lim v_2$.*

Given a time structure $\mathcal{T}$, let $\mathcal{T}^\infty$ be the set of downward closed intervals, i.e. for any $T \in \mathcal{T}^\infty$, $t \in T$ implies that for all $t' \leq t$, $t' \in T$. Obviously $\mathcal{T} \in \mathcal{T}^\infty$. A trace $v : \mathcal{T} \to A$ can be extended to its *completion* $v^\infty : \mathcal{T}^\infty \to A$ as: $v^\infty(T) = \lim v_{|T}$ where $v_{|T}$ denotes the restriction of v onto T. A trace completion provides values at infinite as well as finite time points. For any trace $v : \mathcal{T} \to A$, $v^\infty(\mathcal{T}) = \lim v$ can be considered as the "final" value. For simplicity, we will use v to refer to both v and its completion v^∞ when no ambiguity arises. Furthermore, let $pre(t) = \{t' | t' < t\}$ and $t - \tau = \{t' | t' < t, d(t, t') \geq \tau\}$ for $\tau \in \mathcal{R}^+$. Clearly, $pre(t) \in \mathcal{T}^\infty$ when $t > \mathbf{0}$, $t - \tau \in \mathcal{T}^\infty$ when

$m(t) \geq \tau > 0$, and $pre(t) = t - 0$. A time structure $\mathcal{T}$ is *semi-discrete* iff $\forall t > \mathbf{0}, pre(t)$ has a greatest element; $\mathcal{T}$ is *well-defined* iff $\forall 0 \leq \tau < \bigvee m(\mathcal{T}), \{t | m(t) \leq \tau\}$ has a greatest element.

Trace structures are derived as follows. Given a time structure $\mathcal{T}$ and a domain $\langle A, \leq_A, \tau \rangle$, the *trace space* is a triple $\langle A^{\mathcal{T}}, \leq_{A^{\mathcal{T}}}, \Gamma \rangle$ where $A^{\mathcal{T}}$ is the product set, i.e. the set of *all* functions from $\mathcal{T}$ to A, $\leq_{A^{\mathcal{T}}}$ is the product partial order relation of the partial order relation $\leq_A$ and Γ is the product topology of the derived metric topology τ. For simplicity, we will use $A^{\mathcal{T}}$ to refer to trace space $\langle A^{\mathcal{T}}, \leq_{A^{\mathcal{T}}}, \Gamma \rangle$ when no ambiguity arises.

Given a linear set of traces $V : L \to A^{\mathcal{T}}$, limits and the limit of V are defined as follows. A trace $V^* \in A^{\mathcal{T}}$ is a *limit* of V, written $V \to V^*$, iff V^* is a limit of V in the derived metric topology. Similar to the properties with limits of a linear set of values, the properties of limits of a linear set of traces are as follows.

Proposition 3.5 *If $V : L \to A^{\mathcal{T}}$ for a linear order L and a trace space $A^{\mathcal{T}}$, then*

(1) $V \to V^*$ iff $V(t) \to V^*(t)$ for all $t \in \mathcal{T}$, and

(2) the set of limits $\{V^* | V \to V^*\}$ is a directed subset in $\langle A^{\mathcal{T}}, \leq_{A^{\mathcal{T}}} \rangle$ and has a least upper bound.

Therefore, we can define the limit of a linear set of traces as follows. Let $V : L \to A^{\mathcal{T}}$, the *limit* of V, written $\lim V$, is defined as the least upper bound of the set of limits of V, $\lim V = \bigvee_{A^{\mathcal{T}}} \{V^* | V \to V^*\}$.

Proposition 3.6 *If $V : L \to A^{\mathcal{T}}$, $(\lim V)(t) = \lim V(t), \forall t \in \mathcal{T}$.*

An *event trace* is a special type of trace which is nonintermittent and whose domain is $\overline{\mathcal{B}}$. A nonintermittent trace is defined as follows. A trace $v : \mathcal{T} \to \overline{A}$ is *nonintermittent* iff for any $T \in \mathcal{T}^{\infty}$, $v(T) = \perp_A$ implies that $\forall T' \supset T$, $v(T') = \perp_A$. A trace $v : \mathcal{T} \to \times_I A_i$ is *nonintermittent* iff v_i is nonintermittent for all $i \in I$.

An event trace $e : \mathcal{T} \to \overline{\mathcal{B}}$ with $e \neq \lambda t.\ \perp_{\mathcal{B}}$ generates a sample time structure $\langle \mathcal{T}_e, d_e, \mu_e \rangle$ of $\langle \mathcal{T}, d, \mu \rangle$ where:

- $\mathcal{T}_e \subseteq \mathcal{T}$ is defined as: $\mathcal{T}_e = \{\mathbf{0}\} \cup \{t > \mathbf{0} | e(t) \neq \perp_{\mathcal{B}}, e(t) \neq e(pre(t))\}$, i.e., each transition point of the event trace defines a time point (Figure 5).
- $d_e = d_{|\mathcal{T}_e \times \mathcal{T}_e}$.
- $\forall t \in \mathcal{T}_e$, $\mu_e([\mathbf{0}, t)) = \mu([\mathbf{0}, t))$. Let $T = \{t | e(t) \neq \perp_{\mathcal{B}}\}$. $\mu_e(\mathcal{T}_e) = \mu(T)$.

Proposition 3.7 *The sample time structure generated by any event trace is semi-discrete and well-defined.*

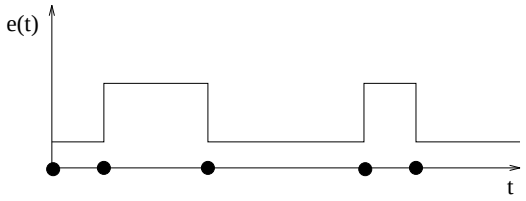

Figure 5: An event trace: each dot depicts a time point

An *event space* is a triple $\langle \mathcal{E}^{\mathcal{T}}, \leq_{\mathcal{E}^{\mathcal{T}}}, \Gamma' \rangle$ where $\mathcal{T}$ is a time structure, $\mathcal{E}^{\mathcal{T}} \subset \overline{\mathcal{B}}^{\mathcal{T}}$ is the set of all event traces on the time structure $\mathcal{T}$, $\leq_{\mathcal{E}^{\mathcal{T}}}$ is the subpartial order relation of the partial order relation $\leq_{\mathcal{B}^{\mathcal{T}}}$ and Γ' is the subspace topology of Γ.

3.4 Transductions

A transduction is a mapping from input traces to output traces which satisfies the causal relationship between its inputs and outputs, i.e. the output value at any time depends only on inputs up to that time. Formally, given $v_1, v_2 \in A^{\mathcal{T}}$ and $\tau \in \mathcal{R}^+$, v_1 and v_2 are *coincident* up to τ iff $\forall t, m(t) \leq \tau \rightarrow v_1(t) = v_2(t)$. A mapping from a trace space to a trace space $F : A^{\mathcal{T}} \rightarrow A'^{\mathcal{T}'}$ is *causal* iff for any $t' \in \mathcal{T}'$, $F(v_1)(t') = F(v_2)(t')$ whenever v_1 and v_2 are coincident up to $m'(t')$. A causal mapping on trace spaces is called a *transduction*. For instance, a state automaton with an initial state defines a transduction on a discrete time structure; a temporal integration with a given initial value is a typical transduction on a continuous time structure. Just as nullary functions represent constants, nullary transductions represent traces. Obviously, transductions are closed under functional composition.

We characterize two classes of transductions: *primitive* transductions and *event-driven* transductions. Primitive transductions are defined on a generic time structure $\mathcal{T}$. *Primitive transductions* are functional compositions of two types of *basic* transductions: transliterations and delays.

A *transliteration* is a pointwise extension of a function. Given $f : A \rightarrow A'$, $f_{\mathcal{T}} : A^{\mathcal{T}} \rightarrow A'^{\mathcal{T}}$ is the pointwise extension of f onto a time structure $\mathcal{T}$: $f_{\mathcal{T}}(v) = \lambda t.f(v(t))$. We will also use f to denote transliteration $f_{\mathcal{T}}$ if no ambiguity arises. Intuitively, a transliteration is a transformational process without memory or internal state, such as a combinational circuit.

Let A be a domain, $v_0 \in A$ be an initial output value and $\mathcal{T}$ be a time structure, a *unit delay* $\delta^A_{\mathcal{T}}(v_0) : A^{\mathcal{T}} \rightarrow A^{\mathcal{T}}$ is a transduction defined as:

$$\delta^A_{\mathcal{T}}(v_0)(v) = \lambda t. \begin{cases} v_0 & \text{if } t = \mathbf{0} \\ v(pre(t)) & \text{otherwise.} \end{cases}$$

A unit delay $\delta^A_{\mathcal{T}}(v_0)$ acts as a unit memory for data in domain A, given a semi-discrete time structure $\mathcal{T}$.

We will use $\delta(v_0)$ to denote unit delay $\delta_{\mathcal{T}}^{A}(v_0)$ if no ambiguity arises. Unit delays are not meaningful for non-semi-discrete time structures. Transport delays are essential for sequential behaviors in dynamic systems. Let A be a domain, $v_0 \in A$ be an initial output value, $\mathcal{T}$ be a time structure and $\tau > 0$ be time delay, a *transport delay* $\Delta_{\mathcal{T}}^{A}(\tau)(v_0) : A^{\mathcal{T}} \to A^{\mathcal{T}}$ is a transduction defined as:

$$\Delta_{\mathcal{T}}^{A}(\tau)(v_0)(v) = \lambda t. \begin{cases} v_0 & \text{if } m(t) < \tau \\ v(t-\tau) & \text{otherwise.} \end{cases}$$

We will use $\Delta(\tau)(v_0)$ to denote transport delay $\Delta_{\mathcal{T}}^{A}(\tau)(v_0)$ if no ambiguity arises.

Primitive transductions are functional compositions of basic transductions, namely transliterations and delays, with all the input/output traces sharing the same time structure. However, a hybrid system consists of components with different time structures. In the rest of this section, we consider event-driven transductions which constitute an important aspect of our model.

First, we consider the types of transductions which map traces of one time structure into those of another. Let $\mathcal{T}_r$ be a reference time of $\mathcal{T}$ with reference time mapping h. The *sample trace* of $v : \mathcal{T}_r \to A$ onto $\mathcal{T}$ is a trace $\underline{v} : \mathcal{T} \to A$, $\underline{v} = \lambda t.v(h(t))$. On the other hand, the *extension trace* of $v : \mathcal{T} \to A$ onto $\mathcal{T}_r$ is a trace $\overline{v} : \mathcal{T}_r \to A$,

$$\overline{v} = \lambda t_r. \begin{cases} v(h^{-1}(t_r)) & \text{if } \exists t \in \mathcal{T}, \mu_r([\mathbf{0}_r, t_r)) \leq \mu([\mathbf{0}, t)) \text{ or } \mu_r([\mathbf{0}_r, t_r)) < \mu(\mathcal{T}) \\ \perp_A & \text{otherwise} \end{cases}$$

where $h^{-1}(t_r) = \{t | h(t) \leq_r t_r\} \in \mathcal{T}^{\infty}$. *Sampling* is a type of transduction whose output is a sample trace of the input. *Extending* is a type of transduction whose output is an extension trace of the input.

An event-driven transduction is a primitive transduction augmented with an extra input, an event trace; it operates at each event point and the output value holds between two events. The additional event trace input of an event-driven transduction is called the *clock* of the transduction. An event-driven transduction works as follows. First, sample the input trace from the reference time $\mathcal{T}$ onto the sample time $\mathcal{T}_e$ generated by the event trace e. Then, perform the transduction in $\mathcal{T}_e$. Finally, extend the output trace from $\mathcal{T}_e$ back to $\mathcal{T}$. Let $\mathcal{E}^{\mathcal{T}}$ be the set of all event traces on time structure $\mathcal{T}$, and $F_{\mathcal{T}} : A^{\mathcal{T}} \to A'^{\mathcal{T}}$ be a primitive transduction. We define an *event-driven transduction* on time structure $\mathcal{T}$ as $F_{\mathcal{T}}^{\circ} : \mathcal{E}^{\mathcal{T}} \times A^{\mathcal{T}} \to A'^{\mathcal{T}}$,

$$F_{\mathcal{T}}^{\circ}(e, v) = \begin{cases} \lambda t.\ \perp_{A'} & \text{if } e = \lambda t.\ \perp_{\mathcal{B}} \\ \\ \overline{F_{\mathcal{T}_e}(\underline{v})} & \text{otherwise.} \end{cases}$$

We will use F° to denote event-driven transduction $F_{\mathcal{T}}^{\circ}$ if no ambiguity arises.

3.5 Dynamics structures

Finally, with preliminaries established, we can characterize the abstract structures of dynamics. Let $\Sigma = \langle S, F \rangle$ be a signature. Given a Σ-domain structure A and a time structure $\mathcal{T}$, a Σ-*dynamics structure* $\mathcal{D}(\mathcal{T}, A)$ is pair $\langle \mathcal{V}, \mathcal{F} \rangle$ where

- $\mathcal{V} = \{A_s^{\mathcal{T}}\}_{s \in S} \cup \mathcal{E}^{\mathcal{T}}$ is an S-sorted set of trace spaces together with the event space;
- $\mathcal{F} = \mathcal{F}_{\mathcal{T}} \cup \mathcal{F}_{\mathcal{T}}^{\circ}$ where $\mathcal{F}_{\mathcal{T}} = \{f_{\mathcal{T}}^{A}\}_{f \in F} \cup \{\delta_{\mathcal{T}}^{A_s}(v_s)\}_{s \in S, v_s \in A_s} \cup \{\Delta_{\mathcal{T}}^{A_s}(\tau)(v_s)\}_{s \in S, \tau > 0, v_s \in A_s}$ is the set of basic transductions, $\mathcal{F}_{\mathcal{T}}^{\circ} = \{F^{\circ} | F \in \mathcal{F}_{\mathcal{T}}\}$ is the set of event-driven transductions.

So far we have presented a topological structure of dynamics by formalizing time, domains and traces in topological spaces and by characterizing primitive and event-driven transductions. With such a topological structure, continuous as well as discrete time and domains can be represented uniformly, and a hybrid dynamic system can be studied in a unitary model.

The advantages of developing an abstract dynamics structure are the following:

- Algebraic specification for the domain structure can be extended to the dynamics structure.
- Algebraic transformation can be applied to control synthesis.
- A real-time programming semantics can be developed on a sound mathematical base.
- A dynamic system can be analyzed at different levels of abstraction in topological spaces.

4 The Semantics of Constraint Nets

So far we have presented the syntactical structure of constraint nets, which is graphical and modular. However, syntax only serves as a mechanism for creating a model, the meaning of the model is not provided. There are many models with syntax similar to constraint nets (Petri nets [23] for example) that have totally different interpretations.

Since transductions are mappings from traces to traces, one direct interpretation of a constraint net is to denote by each location a trace of the right sort. Thus, a constraint net denotes a set of equations with locations as variables and transductions as functions; the semantics of the constraint net is a (the) solution of the set of equations.

Given a set of equations, there are only three possibilities: the set of equations has (1) no solution, (2) exactly one solution and (3) more than one solution. For example, if $x \in \mathcal{R}$, $x = x - 2$ has no solution, $x = 0.5x - 2$ has one solution -4, and $x = x^2 - 2$ has two solutions, -1 and 2. A common

technique is to define the semantics to be the least element in the solution set w.r.t. a partial order. Even though we have defined a partial order for any trace space, it is not guaranteed that the least solution exists for any set of equations and it is not clear how to solve the set of equations.

In this section, we first present fixpoint theorems in partial orders and then examine the properties of a dynamics structure in partial order topologies. With these established, we can define the semantics of constraint nets using fixpoint theorems in partial orders.

4.1 Fixpoint theorems and continuous domain structures

The fixpoint theorems used here are for *complete partial orders* (*cpo*'s). *Continuous functions* are functions which are continuous in partial order topologies.

Theorem 4.1 (Fixpoint Theorem I) *Let A be a cpo. Every continuous function $f : A \to A$ has a least fixpoint $\mu.f$. In particular, $\mu.f = \bigvee_A \{f^n(\perp_A)\}$.*

By extending f to a function of two arguments, we have the following theorem.

Theorem 4.2 (Fixpoint Theorem II) *Let A and A' be two cpo's. If $f : A \times A' \to A'$ is a continuous function, then there exists a unique continuous function $\mu.f : A \to A'$, such that for all $a \in A$, $(\mu.f)(a)$ is the least fixpoint of $f_a : A' \to A'$, where $f_a = \lambda x.f(a, x)$, or equally, $\forall a \in A, (\mu.f)(a) = f(a, (\mu.f)(a))$.*

It is becoming clear that if the partial orders of the set of multi-sorted trace spaces and the event space are *cpo*'s, and the set of basic transductions and their event-driven extensions are continuous in partial order topologies, the semantics of constraint nets composed of primitive and event-driven transductions can be provided and constructed using the fixpoint theorems.

The fact that the partial orders of multi-sorted trace spaces are indeed *cpo*'s is established by the following propositions.

Proposition 4.1 *The partial order of a simple domain is a cpo.*

Proposition 4.2 *The partial order of a domain is a cpo.*

Proposition 4.3 *The partial order of a trace space is a cpo.*

Proposition 4.4 *The partial order of an event space is a cpo.*

The following propositions characterize the continuity of basic transductions in partial order topologies.

Proposition 4.5 *If $f : A \to A'$ is continuous, then for any time structure $\mathcal{T}$, $f_{\mathcal{T}} : A^{\mathcal{T}} \to A'^{\mathcal{T}}$ is continuous.*

Proposition 4.6 *Unit delays are continuous given semi-discrete time structures.*

Proposition 4.7 *Transport delays are continuous given well-defined time structures.*

The following property characterizes the continuity of event-driven transductions.

Proposition 4.8 *If a primitive transduction F is continuous, then the event-driven transduction F° is continuous.*

We conclude this section by introducing continuous domain structures. Let $\Sigma = \langle S, F \rangle$ be a signature. A Σ-domain structure $\langle \{A_s\}_{s \in S}, \{f^A\}_{f \in F} \rangle$ is *continuous* iff f^A is continuous in the partial order topology on A, for all $f \in F$.

In fact, to be continuous on a domain w.r.t. a partial order topology is not a real restriction. Given any partial or total function $f : \times_I A_i \to A$, a continuous function $\bar{f} : \times_I \overline{A_i} \to \overline{A}$ can be defined as:

$$\bar{f}(a) = \begin{cases} f(a) & \text{if } a \in \times_I A_i \text{ and } f(a) \text{ is defined} \\ \bot_A & \text{otherwise.} \end{cases}$$

We call $\bar{f}$ a *strict extension* of function f, or a *strict continuous function.* For example, let $\Sigma_r = \langle r, \{0, +, \cdot\} \rangle$ with $0 :\to r$, $+ : r, r \to r$ and $\cdot : r, r \to r$. Then $\langle \overline{\mathcal{R}}, \{0, \bar{+}, \bar{\cdot}\} \rangle$ is a continuous Σ_r-domain structure, where $+$ and $\cdot$ are addition and multiplication on $\mathcal{R}$. We will also use f to denote its strict extension if no ambiguity arises.

Finally, we come to the theorem on Σ-dynamics structures.

Theorem 4.3 (Continuous Σ-dynamics structure) *If A is a continuous Σ-domain structure and $\mathcal{T}$ is a well-defined time structure, the Σ-dynamics structure $\mathcal{D}(\mathcal{T}, A) = \langle \mathcal{V}, \mathcal{F} \rangle$ satisfies (1) $\mathcal{V}$ is a multi-sorted set of cpo's and (2) all transductions except unit delays in $\mathcal{F}$ are continuous in the partial order topology. If $\mathcal{T}$ is also semi-discrete, all transductions in $\mathcal{F}$ are continuous.*

4.2 Fixpoint semantics of constraint nets

In this section, we come to the semantics of constraint nets composed of primitive transductions and event-driven transductions. Intuitively, a constraint net is a set of equations, with locations as variables and transductions as functions. We take the least fixpoint of the set of equations as the semantics of the net.

Let $\Sigma = \langle S, F \rangle$ be a signature. A constraint net with signature Σ is a triple $CN_\Sigma = \langle Lc, Td, Cn \rangle$,

- each location $l \in Lc$ is associated with a sort $s \in S$, where $c \in S$ is a special sort denoting clocks: the sort of location l is written as s_l;
- each transduction $F \in Td$ is a primitive transduction or an event-driven transduction. There are three types of basic transductions:
  1. transliteration f with a set of input ports I_f and an output port o_f for $f : s^* \to s \in F$ where $s^* : I_f \to S$: the sort of an input port $i \in I_f$ is s^*_i and the sort of the output port is s;
  2. unit delay δ^s with one input port and one output port for $s \in S$, whose input and output ports are associated with sort s;
  3. transport delay Δ^s with one input port and one output port for $s \in S$, whose input and output ports are associated with sort s.

  If F is a primitive transduction, with a set of input ports I_F and an output port o_F, let F° be the event-driven extension of F, with the set of input ports $\{e\} \cup I_F$ and the output port o_F: the sort of the event input port e is c.

Let CN_Σ be a constraint net with signature Σ and $\mathcal{D}(\mathcal{T}, A) = \langle \mathcal{V}, \mathcal{F} \rangle$ be a continuous Σ-dynamics structure. CN_Σ denotes a set of equations $\{o = F_o(\vec{i})\}_{o \in O(CN)}$ where F_o is either a primitive transduction composed of corresponding continuous transductions in $\mathcal{F}$, or an event-driven transduction, $o \in O(CN)$ is the output location of F_o and $\vec{i} : I_{F_o} \to Lc$ is the tuple of input locations of F_o. The semantics of a constraint net is defined using **Fixpoint Theorem II**.

The *semantics* of a constraint net CN_Σ defined on a continuous Σ-dynamics structure is the least fixpoint of the set of equations $\{o = F_o(\vec{i})\}_{o \in O(CN)}$; it is a transduction from the input trace space to the output trace space, i.e. $[\![CN]\!] : \times_{I(CN)} A^{\mathcal{T}}_{s_i} \to \times_{O(CN)} A^{\mathcal{T}}_{s_o}$.

If we consider the semantics of a constraint net as a transduction, then the semantics of a module will be defined as a set of transductions. Given that the semantics of a constraint net CN is $[\![CN]\!] : \times_{I(CN)} A^{\mathcal{T}}_{s_i} \to \times_{O(CN)} A^{\mathcal{T}}_{s_o}$, the *semantics of a module* $CN(I, O)$ is $[\![CN(I,O)]\!] = \{F_u : \times_I A^{\mathcal{T}}_{s_i} \to \times_O A^{\mathcal{T}}_{s_o}\}_{u \in U}$ where $F_u(i) = [\![CN]\!]_{|O}(u, i)$ and $U = \times_{I(CN)-I} A^{\mathcal{T}}_{s_i}$ is the set of hidden input traces. The semantics of a composite module can be derived from the semantics of its components [31].

A module describes a relation between inputs and outputs. If we take the union of the set of transductions in the semantics of module $CN(I, O)$, noting that each transduction is a set of pairs, we obtain a relation on the relation scheme $I \cup O$; each relation tuple is a mapping from input/output

locations to traces, which satisfying all the equations of the net. If we represent the semantics of CN also as a relation on the relation scheme $Lc = I(CN) \cup O(CN)$, noting that functions/transductions are a special type of relations, we have $\bigcup[\![CN(I,O)]\!] = \Pi_{I \cup O}[\![CN]\!]$, where Π_X denotes the projection of a relation onto the relation scheme X. If $I \subset I(CN)$, $\bigcup[\![CN(I,O)]\!]$ is a relation in general, rather than a function/transduction. Therefore, nondeterminism can be modeled via hidden inputs. Two modules $CN(I,O)$ and $CN'(I,O)$ are *nondeterministically equivalent*, written $CN(I,O) \simeq CN'(I,O)$, iff $\bigcup[\![CN(I,O)]\!] = \bigcup[\![CN'(I,O)]\!]$.

Even though every constraint net defined on a continuous dynamics structure has a least fixpoint, the least fixpoint may not be well-defined. For example, $x = 0.5x - 2$, $x \in \overline{\mathcal{R}}$, has $\perp_{\mathcal{R}}$ as its least fixpoint, and -4 as another fixpoint. The relationship among well-definedness of constraint nets, strict continuous functions and algebraic loops has been studied [31].

4.3 Parameterized nets and temporal integration

Finishing up this section, we introduce parameterized nets, a net associated with a set of parameters, then discuss limiting behaviors of parameterized nets, which will be used for providing a semantics of constraint nets with temporal integration.

Many systems share the same structure or follow the same law, while exhibiting different behaviors w.r.t. different parameters. A *parameter* is a variable whose value does not change with time, but differs from system to system, for example, mass, friction coefficient, initial state, time delay, gain, threshold, etc. Let P be a set of parameters, associated with each parameter $p \in P$ is a set of values D_p. CN^P is a *parameterized net* iff CN is a constraint net and P is a set of parameters in CN. The semantics of CN^P, denoted $[\![CN^P]\!]$, is a mapping from the parameter space to the set of transductions, i.e. $[\![CN^P]\!] : \times_P D_p \rightarrow (\times_{I(CN)} A^{\mathcal{T}}_{s_i} \rightarrow \times_{O(CN)} A^{\mathcal{T}}_{s_o})$ such that for any parameter tuple $v \in \times_P D_p$, $[\![CN^P]\!](v) = [\![CN[v/P]]\!]$ where $CN[v/P]$ denotes that each $p \in P$ in CN is replaced by its corresponding value $v(p)$.

Infinitesimals are an important class of parameters for limiting behaviors. Let ϵ be a parameter with $D_\epsilon = (0,1) \subset \mathcal{R}$. Let $\leq_{D_\epsilon}$ be a partial order relation such that $\epsilon_1 \leq_{D_\epsilon} \epsilon_2$ iff $\epsilon_2 \leq_{\mathcal{R}} \epsilon_1$. $\langle D_\epsilon, \leq_{D_\epsilon} \rangle$ is a linear order. We call such a parameter ϵ an *infinitesimal*. The *limiting semantics* of a closed constraint net CN with an infinitesimal ϵ, written $[\![CN^*]\!]$, is defined as the limit of the linear set of traces $[\![CN^\epsilon]\!]$, i.e. $[\![CN^*]\!] = \lim [\![CN^\epsilon]\!]$.

So far we have no definition for temporal integration, the most important type of transduction for continuous time structures. We now define temporal integration on vector spaces and provide the semantics of constraint nets with temporal integration using limiting semantics.

Let U be a topological vector space [29], which is a special class of domain structures, with functions $+ : U \times U \to U$ and $\cdot : \overline{\mathcal{R}} \times U \to U$ continuous in both the partial order and the derived metric topology. A *temporal integration* $\int(s_0) : U^{\mathcal{T}} \to U^{\mathcal{T}}$ with an initial state $s_0 \in U$ can be defined as follows.

First, consider the case where $\mathcal{T}$ is a discrete time structure. Given $\mathcal{T}$ discrete, $pre(t)$ has a greatest element for any $t > \mathbf{0}$, which will also be denoted by $pre(t)$. Temporal integration is as follows:

$$\int(s_0)(u) = \lambda t. \begin{cases} s_0 & \text{if } t = \mathbf{0} \\ \Sigma_{\mathbf{0} < t' \leq t} \mu([pre(t'), t')) \cdot u(pre(t')) & \text{otherwise.} \end{cases}$$

Clearly, the transduction $\int(s_0)$ is the least fixpoint of equation

$$s = \delta(s_0)(s) + dt \cdot \delta(0)(u)$$

where

$$dt = \lambda t. \begin{cases} 0 & \text{if } t = \mathbf{0} \\ \mu([pre(t), t)) & \text{otherwise.} \end{cases}$$

Now given that $\mathcal{T}$ is an arbitrary time structure, let $\mathcal{T}_e$ be a discrete sample time of $\mathcal{T}$, generated by an event trace e. In particular, let e be the solution of $e = \Delta(\epsilon)(0)(\neg e)$ for an infinitesimal ϵ. Let $int_{s_0}(u, s) = \delta(s_0)(s) + dt \cdot \delta(0)(u)$, a temporal integration $\int(s_0)$ corresponds to a module $CN(u, s)$ where CN is represented by following two equations:

$$s = int^{\circ}_{s_0}(e, u, s), e = \Delta(\epsilon)(0)(\neg e)$$

with $\epsilon > 0$ as an infinitesimal. This definition can be considered as derived from the forward Euler method; however, we are interested in semantics, rather than numerical simulation of differential equations.

For example, let us investigate the limiting semantics of the net in Figure 2 with U as $\overline{\mathcal{R}}$, $\mathcal{T}$ as $\mathcal{R}^+$ and $f : \overline{\mathcal{R}} \to \overline{\mathcal{R}}$ where $f = \lambda s.(-s)$. This closed net is represented by three equations:

$$s = int^{\circ}_{s_0}(e, u, s), e = \Delta(\epsilon)(0)(\neg e), u = -s.$$

The solution for e is:

$$e = \lambda t. \begin{cases} 0 & \text{if } \lfloor \frac{t}{\epsilon} \rfloor \text{ is even} \\ 1 & \text{otherwise.} \end{cases}$$

The solution for s can be computed as the least fixpoint of $s = int^{\circ}_{s_0}(e, -s, s)$. According to **Fixpoint Theorem I**, let $s^0 = \lambda t.\ \perp_{\mathcal{R}}$, the least element, we have

$$
\begin{aligned}
s^1 &= int^{\circ}_{s_0}(e, -s^0, s^0) = \lambda t. \begin{cases} s_0 & \text{if } t < \epsilon \\ \perp_{\mathcal{R}} & \text{otherwise,} \end{cases} \\
s^2 &= int^{\circ}_{s_0}(e, -s^1, s^1) = \lambda t. \begin{cases} s_0 & \text{if } t < \epsilon \\ s_0 - \epsilon s_0 & \text{if } \epsilon \le t < 2\epsilon \\ \perp_{\mathcal{R}} & \text{otherwise,} \end{cases} \\
&\vdots \\
s^{k+1} &= int^{\circ}_{s_0}(e, -s^k, s^k) = \lambda t. \begin{cases} s_0 & \text{if } t < \epsilon \\ s_0 - \epsilon s_0 & \text{if } \epsilon \le t < 2\epsilon \\ \vdots & \\ (\Sigma_{i=0}^{k}(-1)^i C_k^i \epsilon^i) s_0 & \text{if } k\epsilon \le t < (k+1)\epsilon \\ \perp_{\mathcal{R}} & \text{otherwise.} \end{cases}
\end{aligned}
$$

Let $s = \bigvee_{\overline{\mathcal{R}}^{\mathcal{R}^+}} \{s^k\}$; s is the least fixpoint of the equation; $s = \lambda t. s^{\lfloor \frac{t}{\epsilon} \rfloor + 1}(t)$. The limiting semantics of the net is $s^* = \lambda t. \lim_{\epsilon \to 0} s(t) = \lambda t. \lim_{\epsilon \to 0} (\Sigma_{i=0}^{k}(-1)^i \frac{k!}{i!(k-i)!} \epsilon^i) s_0$ where $k = \lfloor \frac{t}{\epsilon} \rfloor$, i.e. $s^* = \lambda t. (\Sigma_{i=0}^{\infty}(-1)^i \frac{t^i}{i!}) s_0 = \lambda t. s_0 e^{-t}$, which is the solution of $\dot{s} = -s$.

Using limiting semantics, other forms of temporal integration, such as temporal integration with bounded state/output values or with a reset input, can also be defined [31].

5 Modeling in Constraint Nets

We have presented a formal model, constraint nets (CN), for hybrid dynamic systems: the syntax of CN is graphical and modular, the semantic of CN is denotational and composite. The modular aspect of CN not only provides hierarchical structures of system composition, but also provides a simple and general concept for nondeterminism. The denotational semantics using fixpoint theorems in partial orders provides a rigorous and straightforward interpretation for the meaning of CN. Finally, parameterized nets and temporal integration increase the representation power of CN. As a result, CN can be used to model hybrid discrete/continuous dynamic systems with various event-driven components, while events can be generated in the feedback loop of other computations.

A hybrid dynamic system consists of modules with different time structures, with its domain structure multi-sorted. A typical hybrid domain structure would include a continuous domain, the set of real numbers $\overline{\mathcal{R}}$, and a discrete or finite domain, $\overline{S}$, with associated functions. A typical reference time for a hybrid dynamic system is the set of nonnegative real numbers $\mathcal{R}^+$. Event-driven modules can be associated with different clocks, characterizing different sample time structures generated by event traces. An event trace can be either with fixed sampling rate t_s, generated by $e = \Delta(t_s)(0)(\neg e)$, or created by some discrete event generator, for instance, a transition is generated whenever the system

enters (leaves) a predefined set of states. Multiple event traces can also be combined to generate other event traces. Typical event interactions are "event or", "event and", and "event select" which can be defined in terms of event logics [25]. With event logic modules, asynchronous components can be coordinated.

A typical hybrid system is a robotic system which consists of a robot coupled to a continuous environment, while the robot itself is an integration of a continuous plant with a discrete controller. This hybrid system consists of three subsystems: the plant, the controller and the environment, each of which can be modeled as a constraint net (Figure 6).

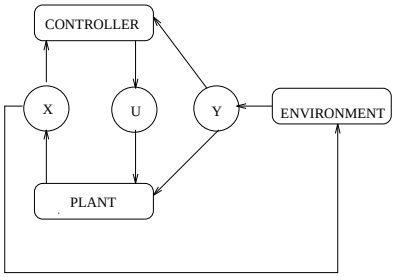

Figure 6: A robotic system where X, U, Y are state, control and sensing variables, respectively

We have been able to model hybrid systems such as an elevator system [33], with an event-driven control structure, and a robot car soccer player [14], with both discrete and continuous components.

Unlike most computational models which are developed on a particular algebra, the constraint net model is an abstraction. However, we have been able to prove that constraint nets can compute any partial recursive function, given a simple domain structure [31]. Traditional analog computation [24] fits this model as well.

Since we define both time and domains as topological spaces, we can study various levels of abstraction via homomorphisms. We have studied time and domain abstraction and refinement, trace and transduction abstraction and equivalence, behavior specification, robustness and complexity within this framework [31].

6 Models for Hybrid Systems

Research on hybrid systems has been carried out for several years. In this section, we survey some typical models for hybrid systems from an historical point of view.

One branch of interest in hybrid system models originated from concurrency models and evolved to timed concurrency models. For example, Alur and Dill [1] developed the theory of timed automata to reason about timed behaviors. Henzinger et al. [12] incorporated time into an interleaving model of concurrency in which upper and lower bounds on time delay are associated with each transition. Various real-time extensions of process algebras [22] have been proposed to model the relative speed of processes. The time Petri net model [5] was introduced to specify and verify real-time systems. None of these models, however, are able to represent continuous change.

Some effort has been made recently to develop models for hybrid systems by generalizing timed transition systems to phase transition systems [15, 21] in which computations consist of alternating phases of discrete transitions and continuous activities. A hybrid system specification using Z was described in [30]. A duration calculus based on continuous time was developed [10] in which integrators can be applied to predicates over a time interval. The use of weakest-precondition predicate transformers in the derivation of sequential, process-control software was discussed in [16].

The constraint net model is more closely related to dataflow-like models and languages, such as the operator net model, LUSTRE, SIGNAL and temporal automata.

The operator net model [2], abstracted from Lucid [28], is defined on continuous algebras using fixpoint theory. The most attractive feature of this model is its independence of any particular algebra. Given a continuous algebra which specifies data types and basic operations, a sequence (continuous) algebra is obtained on which an operator net can be defined. (This idea was used in the development of the constraint net model.) LUSTRE [6], a development based on Lucid, is a real-time programming language, in which sequences are interpreted as time steps. In addition, LUSTRE introduces clocks, so that any expression is evaluated at its clock's sampling rate. However, time structures in LUSTRE are discrete, rather than continuous.

SIGNAL [4, 3], similar to LUSTRE, is a real-time (reactive) programming language. As in LUSTRE, clocks are introduced to trigger various components. Again, the semantics of SIGNAL is based on discrete time structures.

The temporal automaton model [13] is a step towards modeling causal functions in multiple time domains. The temporal automaton model provides explicit representation of process time, symmetric representation of a machine and its environment, and aggregation of individual machines to form a machine at a coarser level of granularity. Even though time can be continuous in this model, there remain untackled problems in modeling continuous change and event control.

A standard hybrid systems modeling language (SHSML) was proposed recently [26]. SHSML is based mostly upon the conceptual definition of a hybrid system that underlies hybrid DSTOOL [9]

and on the modeling and simulation environment provided by SIMNON [7]. A system modeled by SHSML consists of continuous (continuous time and domain, for example, differential equations), discrete (discrete time and continuous domain, for example, difference equations) and logic (discrete time and domain) components. SHSML can be considered as an architecture definition language for software and hardware codesign. However, there is still no formal semantics for this language.

A topological structure for hybrid systems has been studied recently by Nerode and Kohn [20]. Continuity in hybrid systems can be represented via the introduction of small topologies. The topology of domains in the constraint net model has been influenced by this development.

7 Conclusion and Related Work

We have developed a unitary model, constraint nets (CN), for hybrid dynamic systems. In order to make the model general, we have studied abstract time and domain structures, from which abstract dynamics structures are derived. The syntactic structure of the model is graphical and modular, while the semantics is denotational and composite.

In summary, the major contributions of CN are: (1) CN models asynchronous and synchronous components, as well as coordination among components with different time structures; (2) CN supports abstract data types and functions, as well as algebraic specifications; (3) CN provides a programming semantics for the design and analysis of hybrid real-time embedded systems; (4) CN serves as a foundation for the specification and verification of hybrid systems.

While modeling focuses on the underlying structure of a system as well as the organization and coordination of its components, global behaviors of the system are not explicitly represented. We have developed a timed linear temporal logic and timed $\forall$-automata [31] as specification languages [34]. Formal, semi-automatic and automatic verification methods for timed $\forall$-automata are developed by integrating a generalized model checking technique for $\forall$-automata with a generalized stability analysis for dynamic systems [34, 35, 31].

A good design methodology can simplify the verification of a robotic system. Control synthesis and system verification can be coupled via requirement specifications. We have explored a relation between constraint satisfaction and dynamic systems via constraint methods [32], and proposed a systematic approach to control synthesis from requirement specifications [31].

The goal of this research is to provide theoretical underpinnings for robot engineering and the systematic development of real-time, hybrid, embedded control systems.

Acknowledgement

We wish to thank Nick Pippenger, Peter Lawrence and Runping Qi for valuable discussions, and an anonymous referee for constructive comments on the paper. This research was supported by the Natural Sciences and Engineering Research Council and the Institute for Robotics and Intelligent Systems.

References

[1] R. Alur and D. Dill. The theory of timed automata. In J.W. deBakker, C. Huizing, W.P. dePoever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, number 600 in Lecture Notes on Computer Science, pages 45 – 73. Springer-Verlag, 1991.

[2] E. A. Ashcroft. Dataflow and eduction: Data-driven and demand-driven distributed computation. In J. W. deBakker, W.P. deRoever, and G. Rozenberg, editors, *Current Trends in Concurrency*, number 224 in Lecture Notes on Computer Science, pages 1 – 50. Springer-Verlag, 1986.

[3] A. Benveniste and P. LeGuernic. Hybrid dynamical systems theory and the SIGNAL language. *IEEE Transactions on Automatic Control*, 35(5):535 – 546, May 1990.

[4] A. Benveniste, P. LeGuernic, Y. Sorel, and M. Sorine. A denotational theory of synchronous reactive systems. *Information and Computation*, (99):192–230, 1992.

[5] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using Time Petri Nets. *IEEE Transactions on Software Engineering*, 17(3):259 – 273, March 1991.

[6] P. Caspi, D. Pilaud, N. Halbwachs, and J. A. Plaice. LUSTRE: A declarative language for programming synchronous systems. In *ACM Proceedings on Principles of Programming Languages*, pages 178 – 188, 1987.

[7] H. Elmqvist. SIMNON – an interactive simulation program for nonlinear systems. In *Proc. of Simulation 77*, 1977.

[8] M. C. Gemignani. *Elementary Topology*. Dover Publications, Inc., 1990.

[9] J. Guckenheimer and A. Nerode. Simulation for hybrid systems and nonlinear control. In *Proc. IEEE Conference on Decision and Control*, pages 2980–2981, December 1992.

[10] M. R. Hansen and C. Zhou. Semantics and completeness of duration calculus. In J.W. deBakker, C. Huizing, W.P. dePoever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, number 600 in Lecture Notes on Computer Science, pages 209 – 225. Springer-Verlag, 1991.

[11] M. Hennessy. *Algebraic Theory of Processes*. MIT Press, 1988.

[12] T. A. Henzinger, Z. Manna, and A. Pnueli. Timed transition systems. In J.W. deBakker, C. Huizing, W.P. dePoever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, number 600 in Lecture Notes on Computer Science, pages 226–251. Springer-Verlag, 1991.

[13] J. Lavignon and Y. Shoham. Temporal automata. Technical Report STAN-CS-90-1325, Robotics Laboratory, Computer Science Department, Stanford University, Stanford, CA 94305, 1990.

[14] A. K. Mackworth. On seeing robots. In A. Basu and X. Li, editors, *Computer Vision: Systems, Theory and Applications*, pages 1–13. World Scientific Press, Singapore, 1993.

[15] O. Maler, Z. Manna, and A. Pnueli. From timed to hybrid systems. In J.W. deBakker, C. Huizing, W.P. dePoever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, number 600 in Lecture Notes on Computer Science, pages 448 – 484. Springer-Verlag, 1991.

[16] K. Marzullo, F. B. Schneider, and N. Budhiraja. Derivation of sequential, real-time, process-control programs. In A. M. van Tilborg and G. M. Koob, editors, *Foundations of Real-Time Computing: Formal Specifications and Methods*, pages 40 – 54. Kluwer Academic Publishers, 1991.

[17] G. H. Mealy. A method for synthesizing sequential circuits. *Bell Sys. Tech. Journal*, 34:1045 – 1079, 1955.

[18] M. D. Mesarovic and Y. Takahara. *General Systems Theory: Mathematical Foundations*. Academic Press, 1975.

[19] E. F. Moore. Gedanken-experiments on sequential machines. In C.E. Shannon and J. McCarthy, editors, *Automata Studies*. Princeton University Press, 1956.

[20] A. Nerode and W. Kohn. Models for hybrid systems: Automata, topologies, controllability, observability. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, number 736 in Lecture Notes on Computer Science, pages 317 – 356. Springer-Verlag, 1993.

[21] X. Nicollin and J. Sifakis. From ATP to timed graphs and hybrid systems. In J.W. deBakker, C. Huizing, W.P. dePoever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, number 600 in Lecture Notes on Computer Science, pages 549 – 572. Springer-Verlag, 1991.

[22] X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In J.W. deBakker, C. Huizing, W.P. dePoever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, number 600 in Lecture Notes on Computer Science, pages 526 – 548. Springer-Verlag, 1991.

[23] J. L. Peterson. *Petri-Net Theory and the Modeling of Systems*. Prentice-Hall, Inc., Englewood Cliffs, 1981.

[24] C. E. Shannon. Mathematical theory of the differential analyzer. *Journal of Mathematics and Physics*, 20:337 – 354, 1941.

[25] I. E. Sutherland. Micropipeline. *Communication of ACM*, 32(6):720 – 738, June 1989.

[26] J. H. Taylor. Software requirements specification for modeling design, development, and evaluation of distributed, hybrid, intelligent contol. Technical report, Odyssey Research Associates, Ithaca, NY, April 1992.

[27] S. Vickers. *Topology via Logic*. Cambridge University Press, 1989.

[28] W. W. Wadge and E. A. Ashcroft. *Lucid, the dataflow programming language.* Academic Press, 1985.

[29] J. Warga. *Optimal Control of Differential and Functional Equations.* Academic Press, 1972.

[30] W. G. Wood. A specification of the cat and mouse problem. In J.W. deBakker, C. Huizing, W.P. dePoever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, number 600 in Lecture Notes on Computer Science, pages 676 – 686. Springer-Verlag, 1991.

[31] Y. Zhang. A foundation for the design and analysis of robotic systems and behaviors, 1994. PhD thesis, forthcoming.

[32] Y. Zhang and A. K. Mackworth. Constraint programming in constraint nets. In *First Workshop on Principles and Practice of Constraint Programming*, pages 303–312, 1993. A revised version will appear in a book with the same title in MIT Press, 1995.

[33] Y. Zhang and A. K. Mackworth. Design and analysis of embedded real-time systems: An elevator case study. Technical Report 93-4, Department of Computer Science, University of British Columbia, February 1993.

[34] Y. Zhang and A. K. Mackworth. Specification and verification of constraint-based dynamic systems. In *Second Workshop on Principles and Practice of Constraint Programming*, pages 206–215, 1994.

[35] Y. Zhang and A. K. Mackworth. Will the robot do the right thing? In *Proc. Artificial Intelligence 94*, pages 255 – 262, Banff, Alberta, May 1994.

A Mathematical Preliminaries

In this appendix, we summarize the mathematical preliminaries required for this paper, based on [8, 29, 27, 11]. We start with general topologies, and then focus on two special kinds of topologies: partial order topologies and metric topologies. Some related concepts will also be stated.

A.1 General topology

Let X be any set. A collection τ of subsets of X is said to be a *topology* on X if the following axioms are satisfied:

- $X \in \tau$ and $\emptyset \in \tau$.
- If $X_1 \in \tau, X_2 \in \tau$, then $X_1 \cap X_2 \in \tau$.
- If $X_i \in \tau$ for all i, then $\cup_i X_i \in \tau$.

The members of τ are said to be τ-*open* subsets of X, or merely *open* if no ambiguity arises. A subset S of X is *closed* iff $X - S$ is open. $\langle X, \tau \rangle$ is called a *topological space*. We will use X to denote $\langle X, \tau \rangle$ if no ambiguity arises.

A topology τ on X is *trivial* iff $\tau = \{X, \emptyset\}$. A topology τ on X is *discrete* iff $\tau = 2^X$.

A topological space is *separated* if it is the union of two disjoint, non-empty open sets; otherwise, it is *connected*.

Let $\langle X, \tau \rangle$ be a topological space and $x \in X$. A subset $N(x)$ of X is called a *neighborhood* of x iff $x \in N(x)$ and $N(x)$ is τ-open. X is a *Hausdorff space* iff $\forall x_1, x_2, \exists N(x_1), N(x_2)$, such that $N(x_1) \cap N(x_2) = \emptyset$.

Let $\langle X, \tau \rangle$ and $\langle X', \tau' \rangle$ be topological spaces. A function $f : X \to X'$ is *continuous* iff for any τ'-open subsets Y of X', $f^{-1}(Y)$ is τ-open. Clearly, continuous functions are closed under functional composition.

A subset $\mathcal{B}$ of τ is said to be a *basis* for the topology τ iff each member of τ is the union of members of $\mathcal{B}$. A subset $\mathcal{S}$ of τ is said to be a *subbasis* for τ iff the set

$$\mathcal{B} = \{B | B \text{ is the intersection of finitely many members of } \mathcal{S}\}$$

is a basis for τ.

Let $\langle X_i, \tau_i \rangle, i \in I$ be a family of topological spaces, and let $\times_I X_i$ be the product set. Let

$$\mathcal{S} = \{\times_I V_i | V_i = X_i \text{ for all but one } i \in I \text{ and each } V_i \text{ is an open subset of } X_i\}.$$

A topology τ on $\times_I X_i$ is the *product topology* iff $\mathcal{S}$ is a subbasis of τ. $\langle \times_I X_i, \tau \rangle$ is called the *product space* of the spaces $\langle X_i, \tau_i \rangle$. If $X_i = X$ with the same topology for all $i \in I$, $\times_I X_i$ is denoted by X^I.

Let $X' \subset X, \tau' = \{W | W = X' \cap U, U \in \tau\}$. It is easy to check that τ' is a topology on X'; τ' is called the *subspace topology* on X'. $\langle X', \tau' \rangle$ is called the *subspace* of $\langle X, \tau \rangle$.

Two topologies on a set can be compared in the following sense: τ_1 is a *finer* topology than τ_2 iff $\tau_1 \supseteq \tau_2$. The trivial topology is the coarsest and the discrete topology is the finest.

In the next two sections, we will introduce two important types of topologies which are between these two extremes: partial order topologies and metric topologies. Partial order topologies are typical non-Hausdorff topologies and metric topologies are typical Hausdorff topologies.

A.2 Partial order topology

A binary relation on a set is called a *partial order relation* iff it is reflexive, transitive and anti-symmetric. Let X be a set, $\leq_X$ be a partial order relation on X, $\langle X, \leq_X \rangle$ is a *partial order*. X is a *linear order* iff $\forall x_1, x_2 \in X$, either $x_1 \leq_X x_2$ or $x_2 \leq_X x_1$. For any partial order relation $\leq_X$, let $\geq_X$ denote the inverse of $\leq_X$, and let $<_X$ ($>_X$) denote the strict relation of $\leq_X$ ($\geq_X$). We will use X to denote partial order $\langle X, \leq_X \rangle$ if no ambiguity arises.

An element $\perp_X \in X$ is a *least element* iff for all $x \in X$, $\perp_X \leq_X x$. Least elements, if they exist, are unique.

A partial order X is *flat* iff there is a least element $\perp_X$ and for any $x, y \in X$, $x \leq_X y$ implies that either $x = y$ or $x = \perp_X$.

An element $u \in X$ is an *upper (lower) bound* of a subset $Y \subseteq X$ iff for all $y \in Y$, $y \leq_X u$ ($u \leq_X y$); u is a *least upper (greatest lower) bound* of Y iff for any upper (lower) bound v of Y, $u \leq_X v$ ($v \leq_X u$). Least upper (greatest lower) bounds of Y, if they exist, are unique and will be denoted by $\bigvee_X Y$ ($\bigwedge_X Y$).

A subset D of X is *directed* iff $\forall x, y \in D, \exists z \in D, x \leq_X z, y \leq_X z$.

A partial order $\langle X, \leq_X \rangle$ is a *complete partial order* (*cpo*) iff there is a least element $\perp_X \in X$, and every directed subset of X has a least upper bound.

Proposition A.1 *A flat partial order is a cpo.*

Let $\{X_i\}_{i \in I}$ be a set of partial orders. The *product partial order relation* on $X = \times_I X_i$ is defined as $x \leq_X x'$ iff $x_i \leq_{X_i} x'_i$ for all $i \in I$. $\langle X, \leq_X \rangle$ is called the *product partial order* of $\{\langle X_i, \leq_{X_i} \rangle\}_{i \in I}$.

Proposition A.2 *The product partial order of cpo's is a cpo.*

Let X be a partial order. The *subpartial order relation* on $X' \subset X$ is defined as $x_1 \leq_{X'} x_2$ iff $x_1 \leq_X x_2$, and $x_1, x_2 \in X'$. $\langle X', \leq_{X'} \rangle$ is called the *subpartial order* of $\langle X, \leq_X \rangle$.

Given any partial order $\langle X, \leq_X \rangle$, the partial order topology is induced as follows. A subset Y of X is *open* iff (1) Y is upward closed, i.e. $y \in Y$ implies that $\forall z, y \leq_X z \rightarrow z \in Y$, and (2) Y is inaccessible from any directed subset, i.e. if $\bigvee_X D \in Y$, then $\exists x \in D$, such that $x \in Y$. The set of open sets on X forms the *partial order topology*.

A partial order X is *non-trivial* iff $\exists x, x', x <_X x'$.

Proposition A.3 *A non-trivial partial order topological space is non-Hausdorff.*

Proposition A.4 *Let X and X' be cpos. A function $f : X \rightarrow X'$ is continuous in the partial order topology iff (1) for any directed subset D of X, $f(D)$ is a directed subset of X', and (2) $\bigvee_{X'} f(D) = f(\bigvee_X D)$.*

A function $f : X_1 \times X_2 \rightarrow X$ is *right continuous* iff $\forall x_1 \in X_1, \lambda x.f(x_1, x)$ is continuous. A *left continuous* function is defined similarly.

Proposition A.5 *A function $f : X_1 \times X_2 \rightarrow X$ is continuous iff it is both left and right continuous.*

Let $f : X \rightarrow X$ be a function. A point $x \in X$ is a *fixpoint* of f iff $x = f(x)$; x is a *least fixpoint* iff for any fixpoint y of f, $x \leq_X y$. Least fixpoints, if they exist, are unique.

Theorem A.1 *Let $f : X \rightarrow X$ be a continuous function on a cpo X. f has a least fixpoint $\mu.f = \bigvee_X \{f^n(\perp_X)\}$.*

A.3 Metric topology

A function $d : X \times X \to \mathcal{R}^+$ is a *metric* on X iff

- $d(x, y) = d(y, x)$.
- $d(x, y) \leq d(x, z) + d(z, y)$.
- $d(x, y) = 0$ iff $x = y$.

$\langle X, d \rangle$ is a *metric space*. A subbasis of the metric topology is $\{N^\epsilon(x)\}_{\epsilon>0, x\in X}$ where $N^\epsilon(x)$ denotes the ϵ-neighborhood of x, i.e. $N^\epsilon(x) = \{x' | d(x, x') < \epsilon\}$.

Proposition A.6 *Metric topologies are Hausdorff.*

A family σ of subsets of X is a *σ-field* iff it contains the empty set, the complement in X of every element in σ and the union of every denumerable subcollection. A function $\mu : \sigma \to \mathcal{R}$ is a *measure* iff $\mu(\emptyset) = 0$ and $\mu(\cup_{j=1}^{\infty} X_j) = \Sigma_{j=1}^{\infty} \mu(X_j)$ if $X_i \cap X_j = \emptyset$ for any $i \neq j$. If $\langle X, \tau \rangle$ is a topological space, then the smallest σ-field containing τ is called the *Borel field of sets*, and denoted by $\Sigma_{Borel}(X)$. A measure defined on $\Sigma_{Borel}(X)$ is called a *Borel measure*.

A.4 Limit

Let L be a linear order, X be a topological space, and $v : L \to X$ be a mapping. A point $v^* \in X$ is a *limit* of v, written $v \to v^*$, iff $\forall N(v^*), \exists l_0, \forall l \geq_L l_0, v(l) \in N(v^*)$.

Proposition A.7 *If $v : L \to X$ and L has a greatest element l_0, $v \to v(l_0)$.*

Proposition A.8 *If X is Hausdorff and $v : L \to X$, then $v \to v_1^*$ and $v \to v_2^*$ imply that $v_1^* = v_2^*$.*

Proposition A.9 *Let L be a linear order, $v : L \to \times_I X_i$. If $\times_I X_i$ is with the product topology, then $v \to v^*$ iff for all $i \in I$, $v_i \to v_i^*$.*

Let X be a metric space. A sequence $v : \mathcal{N} \to X$ is *Cauchy* iff $d(v(n), v(m)) \to 0$ when $n, m \to \infty$.

A.5 Vector space

A *vector space* is a set X for which are defined the functions: $+ : X \times X \to X$ and $\cdot : \mathcal{R} \times X \to X$ and with an element $0_X \in X$ satisfying the following conditions:

$$x + y = y + x, (x + y) + z = x + (y + z),$$

$$\alpha(x + y) = \alpha x + \alpha y, (\alpha + \beta)x = \alpha x + \beta x,$$

$$\alpha(\beta x) = (\alpha\beta)x, x + 0_X = x, 0x = 0_X, 1x = x.$$

Let $\Sigma_I x_i$ denote the sum of all elements $x_i, i \in I$. A *topological vector space* is a vector space with a topology such that $+$ and $\cdot$ are continuous functions.

B Proofs of Theorems and Propositions

Proof of Proposition 3.1: (1) The only neighborhood of $\perp_A$ is $\overline{A}$, so $v(l) \in N(\perp_A)$ for all l. (2) If $v_1^* \neq v_2^*$, one of them must be $\perp_A$, since A is Hausdorff with unique limits (Propositions A.6 and A.8).

Proof of Proposition 3.2: (1) Follows from Proposition A.9. (2) For any two limits v and v', the least upper bound of v and v' is also a limit.

Proof of Proposition 3.3: $(\bigvee_A D)_i = \bigvee_{A_i} D_i$ where $D_i = \Pi_i D$.

Proof of Proposition 3.4: If A is flat, $v_1^* \neq \perp_A$ implies that $v_2^* \neq \perp_A$. If A is a product, $\lim v_{1i}^* \leq_{A_i} \lim v_{2i}^*$. Therefore $\lim v_1^* \leq_A \lim v_2^*$.

Proofs of Propositions 3.5 and 3.6: Proofs are similar to Proposition 3.2, 3.3.

Proof of Proposition 3.7: If $\mathcal{T}_e$ is not semi-discrete, there is $t \in \mathcal{T}_e$, $pre(t) \subset \mathcal{T}_e$ has no greatest element, i.e. for any $t' <_e t$, there is $t'', t' <_e t'' <_e t$, which means there is infinitely many transitions between t' and t, then $e(pre(t))$ will not be defined. Similarly, $\mathcal{T}_e$ is well-defined.

Proofs of Propositions 4.1 – 4.3: Follows from Propositions A.1 and A.2.

Proof of Proposition 4.4: Let $\mathcal{V} \subset A^{\mathcal{T}}$ be the set of nonintermittent traces. The least element in $\mathcal{V}$ is $\lambda t.\ \perp_A$. The least upper bound of a directed subset D of $\mathcal{V}$ is $\bigvee_{\mathcal{V}} D = \lambda t. \bigvee_A D(t)$ which is also in $\mathcal{V}$ since $(\bigvee_{A^{\mathcal{T}}} D)(T) \geq_A \bigvee_A D(T)$ (Proposition 3.4), if $(\bigvee_{A^{\mathcal{T}}} D)(T)$ is $\perp_A$, $d(T)$ is $\perp_A$ for all $d \in D$.

Proof of Proposition 4.5: Let $D \subseteq A^{\mathcal{T}}$ be directed and v^* be the least upper bound of D. We will prove that $f_{\mathcal{T}}(\bigvee_{A^{\mathcal{T}}} D) = \bigvee_{A'^{\mathcal{T}}} f_{\mathcal{T}}(D)$, i.e. for any t, $f_{\mathcal{T}}(v^*)(t) = (\bigvee_{A'^{\mathcal{T}}} f_{\mathcal{T}}(D))(t)$.

$$\begin{aligned} f_{\mathcal{T}}(v^*)(t) &= f(v^*(t)) = f(\bigvee_A \{v(t) | v \in D\}) \\ &= \bigvee_{A'} \{f(v(t)) | v \in D\} \quad \text{since } f \text{ is continuous} \\ &= \bigvee_{A'} \{f_{\mathcal{T}}(v)(t) | v \in D\} = \bigvee_{A'^{\mathcal{T}}} f_{\mathcal{T}}(D)(t). \end{aligned}$$

Proof of Proposition 4.6: For any unit delay $\delta_{\mathcal{T}}^A(v_0) : A^{\mathcal{T}} \to A^{\mathcal{T}}$, let $D \subseteq A^{\mathcal{T}}$ be directed and v^* be the least upper bound of D. Since $\mathcal{T}$ is semi-discrete, $pre(t)$ has a greatest element, which will also be denoted by $pre(t)$.

$$\begin{aligned} \delta_{\mathcal{T}}^A(v_0)(v^*)(t) &= \begin{cases} v_0 & \text{if } t = \mathbf{0} \\ v^*(pre(t)) = \bigvee_A \{v(pre(t)) | v \in D\} & \text{otherwise} \end{cases} \\ &= \bigvee_A \{\delta_{\mathcal{T}}^A(v_0)(v)(t) | v \in D\} \\ &= (\bigvee_{A^{\mathcal{T}}} \delta_{\mathcal{T}}^A(v_0)(D))(t). \end{aligned}$$

Proof of Proposition 4.7: Since $\mathcal{T}$ is well-defined, $t - \tau$ has a greatest element when $m(t) \geq \tau$.

Proof of Proposition 4.8: First we prove sampling and extending are continuous. Let $\mathcal{T}$ be a time structure and $\mathcal{T}_r$ be a reference time structure of $\mathcal{T}$ with a reference time mapping h. Sampling is a transduction $S_{\mathcal{T},\mathcal{T}_r} : A^{\mathcal{T}_r} \to A^{\mathcal{T}}$. We prove that it is continuous.

Let $D \subseteq A^{\mathcal{T}_r}$ be directed and v^* be the least upper bound of D. Let $\underline{v}$ be $S_{\mathcal{T},\mathcal{T}_r}(v)$.

$$\underline{v^*}(t) = v^*(h(t)) = \bigvee_A \{v(h(t)) | v \in D\} = \bigvee_A \{\underline{v}(t) | v \in D\} = (\bigvee_{A^{\mathcal{T}}} \{\underline{v} | v \in D\})(t).$$

Therefore $\underline{\bigvee_{A^{\mathcal{T}_r}} D} = \bigvee_{A^{\mathcal{T}}} \underline{D}$.

Similarly, extending is continuous, given the sample time structure well-defined.

The proof is divided into two steps. First, F° is right continuous if F is continuous. Second, F° is left continuous. Therefore, according to Proposition A.5, F° is continuous.

For any time structure $\mathcal{T}$ and any fixed event trace e, sampling to $\mathcal{T}_e$ is continuous and extending to $\mathcal{T}$ is also continuous. If F is continuous, F° is right continuous since continuous functions are closed under functional composition.

Now we prove that it is left continuous. Let $\mathcal{T}$ be any time structure and $v \in A^{\mathcal{T}}$ be fixed. For any directed subset D of $\mathcal{E}^{\mathcal{T}}$, D is a chain. According to the definition, $F^\circ_{\mathcal{T}}(D, v)$ is a chain too, i.e. a directed subset. Furthermore for any t if $(\bigvee_{\mathcal{E}^{\mathcal{T}}} D)(t) \neq \perp_{\mathcal{B}}$, there is $d \in D$ such that for all $t' \leq t, d(t') = (\bigvee_{\mathcal{E}^{\mathcal{T}}} D)(t')$, i.e. $\bigvee_{A'^{\mathcal{T}}} F^\circ_{\mathcal{T}}(D, v) \geq F^\circ_{\mathcal{T}}(\bigvee_{\mathcal{E}^{\mathcal{T}}} D, v)$. On the other hand, $F^\circ_{\mathcal{T}}$ is monotonic w.r.t. the left argument, i.e., $\bigvee_{A'^{\mathcal{T}}} F^\circ_{\mathcal{T}}(D, v) \leq F^\circ_{\mathcal{T}}(\bigvee_{\mathcal{E}^{\mathcal{T}}} D, v)$. Therefore $\bigvee_{A'^{\mathcal{T}}} F^\circ_{\mathcal{T}}(D, v) = F^\circ_{\mathcal{T}}(\bigvee_{\mathcal{E}^{\mathcal{T}}} D, v)$, it is left continuous.

Proof of Theorem 4.1: By Theorem A.1.

Proof of Theorem 4.2: Let $F^0(a) = f(a, \perp_{A'})$ and $F^{k+1}(a) = f(a, F^k(a))$. Since f is continuous, it is right continuous. A continuous function in any partial order is also monotonic. Therefore,

$$F^0(a) \leq_{A'} F^1(a) \leq_{A'} F^2(a) \ldots \leq_{A'} F^k(a) \leq \ldots.$$

Let $\mu.f(a) = \bigvee_{A'} \{F^k(a) | k \geq 0\}$. Clearly $\mu.f(a)$ is the least fixpoint of $f_a : A' \to A'$.

Next we prove that $\mu.f$ is continuous. Clearly for every k, F^k is continuous since f is continuous and continuity is closed under functional composition. Therefore, for any directed subset D of A,

$$\begin{aligned}
\mu.f(\bigvee_A D) &= \bigvee_{A'} \{F^k(\bigvee_A D) | k \geq 0\} \\
&= \bigvee_{A'} \{\bigvee_{A'} \{F^k(D)\} | k \geq 0\} \\
&= \bigvee_{A'} \{\bigvee_{A'} \{F^k(a) | k \geq 0\} | a \in D\} \\
&= \bigvee_{A'} \mu.f(D).
\end{aligned}$$

Proof of Theorem 4.3: Follows from Propositions 4.1–4.8.