BC CTRL Control Language

A prose only instruction set for pen and paper control of complex integer driven systems, with all coefficients permitted

Purpose

This document defines a compact control language whose semantics are grounded in integer arithmetic, integer linear combinations, and composable state update operators. The language is intended to be usable by hand. Each construct is specified using words only, while remaining mathematically precise. All coefficients are permitted, meaning coefficients may be negative, zero, or positive integers. The core idea is to represent control actions as objects that can be combined, simplified, and iterated without ambiguity.

Foundational domains

The language uses the integers as its fundamental coefficient domain. A scalar is a single integer. A state is an ordered list of integers of fixed length, also called an integer vector. A linear transform is an integer matrix, meaning a rectangular array of integers that acts on an integer vector by the standard matrix vector multiplication rule. A basis is an ordered finite list of integers. A coefficient list is an ordered list of integers of the same length as its basis.

Basis coefficient evaluation

Given a basis and a matching coefficient list, the basis coefficient value is defined to be the integer obtained by multiplying each coefficient by the corresponding basis element and summing the results. This is the language's primary scalar extraction operation. It provides a controllable way to encode, decode, and steer integer quantities using a chosen basis and freely selected integer coefficients.

Span semantics and greatest common divisor

For a fixed basis, consider the set of all integers that can be produced as a basis coefficient value when the coefficients range over all integer lists of the appropriate length. This set is exactly the set of all integer multiples of the greatest common divisor of the basis elements. Consequently, a basis generates all integers if and only if the greatest common divisor of the basis elements is one. This fact provides a correctness criterion for whether a chosen basis has full reach over the integers.

State operators as affine integer transforms

A state operator is an action on integer states. An operator consists of an integer matrix together with an integer offset vector of matching length. Applying the operator to a state produces a new state by first multiplying the matrix by the current state and then adding the offset vector componentwise. This construction is an affine transform over the integers. It is the language's primary representation of a control step in a multivariate system.

Bounded operators for anti chaos control

To prevent uncontrolled growth during manual calculation, an operator may additionally carry a modulus specification. The modulus may be a single positive integer applied uniformly to every state component, or a vector of positive integers applied componentwise. A bounded operator is applied by performing the affine update and then reducing each resulting component modulo its corresponding modulus. This produces a wrapped, bounded evolution that keeps states within a fixed residue range and enables reliable hand execution.

Operator composition and action compression

Two operators can be composed to form a single operator that has exactly the same effect as applying the first operator and then the second. The composed operator's matrix part is the matrix product obtained by multiplying the second matrix by the first matrix. The composed operator's offset is obtained by applying the second matrix to the first offset and then adding the second offset. This composition law turns sequences of actions into a calculable algebra. In practice it enables action compression, verification of long pipelines, and equivalence checking between two control strategies.

Iteration and time indexed evolution

A system evolution is a sequence of states indexed by discrete time steps. Each step is produced by applying a selected operator to the current state. When the same operator is used at every step, the evolution is called autonomous and the state after a given number of steps is obtained by iterating that operator the corresponding number of times. When operators may vary by step, the state after a span of steps is obtained by composing the operators in the order they are applied and then applying the resulting composed operator to the starting state.

Optional polynomial representation as a basis instance

Polynomials with integer coefficients can be represented in the same basis coefficient spirit by treating the descending powers of an indeterminate as a monomial basis. A polynomial is then specified by an ordered coefficient list, interpreted against that monomial basis to yield a polynomial function. This aligns the language's scalar basis coefficient operation with common algebraic practice while preserving the principle that the coefficient domain is the integers.

Minimal paper protocol

A minimal manual log records, for each time step, the current state, the operator used, and the resulting next state. For bounded dynamics, the modulus specification is recorded as part of the operator. For scalar control or measurement, the log records the chosen basis, the chosen coefficient list, and the produced integer value. This protocol is sufficient to define a system, execute it stepwise, compress and compare action sequences, enforce bounds, and reason about growth, periodicity, and stability within an integer state space.

Implementation note: Although this document avoids symbolic formulas, each sentence is intended to correspond to a standard algebraic definition over the integers, with matrix multiplication, vector addition, greatest common divisor, and modular reduction interpreted in their conventional mathematical senses.