Extending Factor, Signal Flow, Control Flow, Dataflow, and PERT/CPM Graphs to Hypergraphs and Superhypergraphs

Takaaki Fujita [1] *

[1] Independent Researcher, Shinjuku, Shinjuku-ku, Tokyo, Japan.
Email: Takaaki.fujita060@gmail.com

Abstract

Graph theory investigates how vertices and edges model relationships and connectivity [1, 2]. Hypergraphs extend this paradigm by permitting hyperedges that join any number of vertices [3], and superhypergraphs further generalize via iterated powerset constructions to capture nested, hierarchical linkages among edges [4,5]. Graphs are ubiquitous across many disciplines. In this paper, we show how five classical graph models—factor graphs, signal flow graphs, control flow graphs, dataflow graphs, and PERT/CPM graphs—can be lifted into the HyperGraph and SuperHyperGraph frameworks, and we present illustrative examples for each. These extensions open the door to systematic, multi-level modeling of complex applications.

Keywords: Superhypergraph, Hypergraph, Factor Graph, Signal Flow Graph, Control Flow Graph, Dataflow Graph, PERT/CPM Graph

Mathematics Subject Classification (2010): 05C65 — Hypergraphs

1 Introduction

1.1 From Graphs to SuperHyperGraphs

Traditional graphs capture pairwise relations via vertices and edges [6]. Hypergraphs generalize this by allowing each *hyperedge* to join arbitrarily many vertices, thus modeling higher-order interactions [7–9]. SuperHyperGraphs take this idea further: by iteratively applying the powerset construction, edges at one level become vertices at the next, producing a nested, multi-level network that naturally encodes hierarchical dependencies [10–14]. This layered framework offers a unified method for representing and analyzing complex systems with recursive or embedded connections [15].

1.2 Graph Paradigms Across Disciplines

Graph-based models are fundamental in many areas. In this work, we concentrate on five canonical structures:

- **Factor Graph:** A bipartite network linking variable nodes to factor nodes, decomposing a global function into a product of local factors [16–20].
- **Signal Flow Graph:** A directed graph whose nodes represent signal points and whose weighted edges model gains, used to derive system transfer functions [21–24].
- **Control Flow Graph:** A directed graph of basic blocks (nodes) and control-transfer edges, central to program analysis, optimization, and testing [25–27].
- **Dataflow Graph:** A directed graph in which vertices are computational actors and edges are token channels, capturing data dependencies for parallel scheduling and implementation [28–31].
- **PERT/CPM Graph:** A directed acyclic graph with events as vertices and activities as edges, enabling critical-path analysis and project scheduling [32].

1.3 Our Contributions

We demonstrate how these five classical graph models can be systematically embedded into the HyperGraph and SuperHyperGraph formalisms. For each model, we provide concrete examples that illustrate the added expressive power of hyperedges and supervertices. This unified approach lays the groundwork for structured, multi-level modeling of complex applications.

2 Preliminaries

Throughout this paper, we work with finite, simple graphs unless stated otherwise. This section reviews the key definitions and notation that will be used in later sections; for more in-depth discussions, please refer to the cited sources.

2.1 Hypergraphs and SuperHypergraphs

Hypergraphs extend ordinary graphs by allowing hyperedges to join any number of vertices simultaneously [3, 33–36]. SuperHyperGraphs build on this by applying iterated powerset constructions to model hierarchical connections among edges [5, 11, 14, 37–39]. These frameworks are employed in molecular modeling, network analysis, and signal processing [40–42].

Definition 2.1 (Base Set)**.** Let S be a nonempty finite set, called the *base set*. All further constructions are drawn from S.

Definition 2.2 (Powerset)**.** The *powerset* of S, denoted $\mathcal{P}(S)$, is the collection of all subsets of S, including the empty set:

$$\mathcal{P}(S) \;=\; \{\, A \mid A \subseteq S \}.$$

Definition 2.3 (Hypergraph)**.** [3, 43] A *hypergraph* $H = (V, E)$ consists of

- a finite vertex set V,
- a finite family $E \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$ of nonempty subsets of V, called *hyperedges*.

Hypergraphs generalize ordinary graphs by permitting edges to join any number of vertices simultaneously.

Definition 2.4 (Iterated Powerset)**.** [44–47] For $k \geq 0$, define

$$\mathcal{P}^0(S) = S, \quad \mathcal{P}^{k+1}(S) = \mathcal{P}\big(\mathcal{P}^k(S)\big).$$

Thus $\mathcal{P}^k(S)$ is the k-fold application of the powerset operator.

Definition 2.5 (n-SuperHyperGraph)**.** [12, 48, 49] Let $n \geq 1$ and S be a base set. An *n-superhypergraph* is a pair

$$\mathrm{SHG}^{(n)} = (V, E), \quad V, E \subseteq \mathcal{P}^n(S),$$

where each element of V is called an *n-supervertex* and each element of E an *n-superedge*. Informally, superhypergraphs capture hierarchical relationships by iterating the hypergraph construction.

Example 2.6 (Corporate Hierarchy as a 3-SuperHyperGraph)**.** Let the base set $S = V_0$ be the eight employees:

$$V_0 = \{\text{Alice}, \text{Bob}, \text{Carol}, \text{Dave}, \text{Eve}, \text{Frank}, \text{Grace}, \text{Hank}\}.$$

Level 1 (Teams) Form three project teams:

$$T_{\text{dev}} = \{\text{Alice}, \text{Bob}, \text{Carol}\}, \quad T_{\text{ops}} = \{\text{Dave}, \text{Eve}, \text{Frank}\}, \quad T_{\text{sales}} = \{\text{Grace}, \text{Hank}\}.$$

Thus

$$V_1 = \{\, T_{\text{dev}},\, T_{\text{ops}},\, T_{\text{sales}} \,\} \;\subseteq\; \mathcal{P}(V_0).$$

Level 2 (Departments) Group teams into departments:

$$D_{\text{eng}} = \{T_{\text{dev}}, T_{\text{ops}}\}, \quad D_{\text{mkt}} = \{T_{\text{sales}}\},$$

$$V_2 = \{\, D_{\text{eng}},\, D_{\text{mkt}} \,\} \;\subseteq\; \mathcal{P}^2(V_0).$$

Level 3 (Divisions) Define divisions as sets of departments:

$$\Delta_{\text{north}} = \{D_{\text{eng}}\}, \quad \Delta_{\text{south}} = \{D_{\text{mkt}}\}, \quad \Delta_{\text{all}} = \{D_{\text{eng}}, D_{\text{mkt}}\},$$
$$V_3 = \{\, \Delta_{\text{north}}, \Delta_{\text{south}}, \Delta_{\text{all}} \,\} \subseteq \mathcal{P}^3(V_0).$$

Superedges At the top level, the company-wide coordination depends on both divisions:

$$E_3 = \{\, \Delta_{\text{all}} \,\} \subseteq \mathcal{P}(V_3).$$

Then

$$\text{SHG}^{(3)} = (V_3, E_3)$$

is a 3-superhypergraph (Definition 2.5) encoding the four-tier corporate structure: employees → teams → departments → divisions.

3 Factor graphs and Their Extensions

A factor graph is a bipartite representation connecting variables and factor nodes to express global functions as product of factors(cf. [16–20]). This framework is extended using Hypergraphs and SuperHypergraphs to model more complex and hierarchical activity dependencies.

Definition 3.1 (Factor Graph)**.** (cf. [16–20]) A *factor graph* is a bipartite graph

$$G = (V, F, E), \quad E \subseteq V \times F,$$

together with a collection of *factor functions* $\{\varphi_f : \mathcal{X}_{N(f)} \to \mathbb{R}_{\geq 0}\}_{f \in F}$, where

- V is a finite set of *variable nodes*,
- F is a finite set of *factor nodes*,
- for each $f \in F$, its *scope* or *neighbor set* $N(f) = \{\, v \in V : (v, f) \in E\}$ is nonempty,
- $\mathcal{X}_S = \prod_{v \in S} \mathcal{X}_v$ is the Cartesian product of the domains $\mathcal{X}_v$ of the variables v.

The factor graph encodes a global nonnegative function

$$\Phi(x) = \prod_{f \in F} \varphi_f\big(x_{N(f)}\big), \quad x \in \mathcal{X}_V.$$

Example 3.2 (3-Bit Parity Code as a Factor Graph)**.** Let $V = \{x_1, x_2, x_3, x_4\}$ be binary variables. We impose two parity-check factors:

$$F = \{f_1, f_2\}, \quad N(f_1) = \{x_1, x_2, x_3\}, \quad N(f_2) = \{x_2, x_3, x_4\}.$$

Define factor functions

$$\varphi_{f_i}(x_{N(f_i)}) = \begin{cases} 1, & \sum_{v \in N(f_i)} x_v \equiv 0 \pmod 2, \\ 0, & \text{otherwise}, \end{cases} \qquad i = 1, 2.$$

The bipartite graph $G = (V, F, E)$ has edges (x, v) whenever $v \in N(f)$. Its global function $\Phi(x) = \varphi_{f_1}(x_1, x_2, x_3)\, \varphi_{f_2}(x_2, x_3, x_4)$ encodes the code constraints.

Definition 3.3 (Factor Hypergraph)**.** A *factor hypergraph* is a pair

$$H = (V, \mathcal{E}), \quad \mathcal{E} \subseteq \mathcal{P}(V) \setminus \{\emptyset\},$$

together with factor functions $\{\psi_e : \mathcal{X}_e \to \mathbb{R}_{\geq 0}\}_{e \in \mathcal{E}}$, where each hyperedge e is a nonempty subset of V, and $\mathcal{X}_e = \prod_{v \in e} \mathcal{X}_v$. It represents the global function

$$\Psi(x) = \prod_{e \in \mathcal{E}} \psi_e\big(x_e\big), \quad x \in \mathcal{X}_V.$$

Example 3.4 (3-Bit Parity Code as a Factor Hypergraph)**.** Using the same variables $V = \{x_1, x_2, x_3, x_4\}$, form the hyperedges

$$\mathcal{E} = \{\, e_1, e_2\}, \quad e_1 = \{x_1, x_2, x_3\}, \quad e_2 = \{x_2, x_3, x_4\}.$$

Assign factor functions $\psi_{e_i} = \varphi_{f_i}$ from Example 3.2. The data structure

$$H = (V, \mathcal{E}, \{\psi_{e_1}, \psi_{e_2}\})$$

is a factor hypergraph whose global function $\Psi(x) = \psi_{e_1}(x_{e_1})\, \psi_{e_2}(x_{e_2})$ matches $\Phi(x)$.

Theorem 3.5 (Factor Hypergraphs Are Hypergraphs)**.** *Let $H = (V, \mathcal{E}, \{\psi_e\})$ be a factor hypergraph. Then $\big(V, \mathcal{E}\big)$ is an (undirected) hypergraph in the sense of Definition 2.3: each $e \in \mathcal{E}$ is a nonempty subset of V.*

Proof. By Definition 3.3, each hyperedge $e \in \mathcal{E}$ satisfies $e \neq \emptyset$ and $e \subseteq V$. Thus $\mathcal{E} \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$, so $(V, \mathcal{E})$ meets the requirements of an undirected hypergraph. □

Theorem 3.6 (Generalization of Factor Graphs)**.** *Every factor graph $G = (V, F, E, \{\varphi_f\}_{f \in F})$ induces a factor hypergraph $H = (V, \mathcal{E}, \{\psi_e\}_{e \in \mathcal{E}})$ by setting*

$$\mathcal{E} \;=\; \big\{\, N(f) \subseteq V : f \in F \big\}, \quad \psi_{N(f)} \;=\; \varphi_f.$$

Under this construction, the global function Φ of Definition 3.1 coincides with the global function Ψ of Definition 3.3.

Proof. Since each factor node $f \in F$ has a nonempty neighbor set $N(f) \subseteq V$, the collection $\mathcal{E} = \{N(f) : f \in F\}$ is a family of nonempty subsets of V. Defining $\psi_{N(f)} = \varphi_f$ makes $\Psi(x) = \prod_{e \in \mathcal{E}} \psi_e(x_e)$ equal to $\prod_{f \in F} \varphi_f(x_{N(f)}) = \Phi(x)$. Thus H is a factor hypergraph whose factorization exactly reproduces that of G, so factor hypergraphs strictly generalize factor graphs. □

Definition 3.7 (Factor n-SuperHypergraph)**.** Let V_0 be a finite set of *atomic variables*, each $v \in V_0$ taking values in a finite domain $\mathcal{X}_v$. For $k \geq 0$, set

$$\mathcal{P}^0(V_0) = V_0, \quad \mathcal{P}^{k+1}(V_0) = \mathcal{P}\big(\mathcal{P}^k(V_0)\big).$$

An *(undirected) factor n-superhypergraph* is a triple

$$\mathcal{F}^{(n)} \;=\; \big(V_n,\; E_n,\; \{\psi_e\}_{e \in E_n}\big)$$

where

- $V_n \subseteq \mathcal{P}^n(V_0)$ is the set of n-supervertices,
- $E_n \subseteq \mathcal{P}(V_n) \setminus \{\emptyset\}$ is the set of n-superedges,
- for each $e \in E_n$, $\psi_e \colon \displaystyle\prod_{u \in e} \mathcal{X}_u \;\longrightarrow\; \mathbb{R}_{\geq 0}$ is a nonnegative *factor function*.

The associated *global function* is

$$\Psi(x) \;=\; \prod_{e \in E_n} \psi_e\big(x_e\big), \quad x \in \prod_{u \in V_n} \mathcal{X}_u,$$

where $x_e = (x_u)_{u \in e}$.

Example 3.8 (3-Bit Parity Code as a Factor 1-SuperHypergraph)**.** Group the two hyperedges into one superactivity. Define

$$V_1 = \{\, e_1, e_2\} \;\subseteq\; \mathcal{P}(V), \quad E_1 = \{\, E\}, \quad E = \{e_1, e_2\} \subseteq V_1.$$

Introduce the super-factor

$$\psi_E\big(x_{e_1}, x_{e_2}\big) \;=\; \psi_{e_1}(x_{x_1,x_2,x_3})\; \psi_{e_2}(x_{x_2,x_3,x_4}).$$

Then $\mathcal{F}^{(1)} = (V_1, E_1, \{\psi_E\})$ is a factor 1-superhypergraph in which a single superedge E enforces both parity checks simultaneously.

Theorem 3.9 (Factor n-SuperHypergraphs Are n-SuperHyperGraphs)**.** *If $\mathcal{F}^{(n)} = (V_n, E_n, \{\psi_e\})$ is a factor n-superhypergraph as in Definition 3.7, then (V_n, E_n) is an n-superhypergraph.*

Proof. By hypothesis $V_n \subseteq \mathcal{P}^n(V_0)$ and $E_n \subseteq \mathcal{P}(V_n)$. Hence

$$V_n \subseteq \mathcal{P}^n(V_0), \quad E_n \subseteq \mathcal{P}(V_n) = \mathcal{P}^{n+1}(V_0).$$

This exactly matches the requirement for an $(n+1)$-superhypergraph in Definition 2.5, and in particular (V_n, E_n) is an n-superhypergraph when one views its superedges as lying in $\mathcal{P}^n(V_0)$. Therefore $\mathcal{F}^{(n)}$ underlies a valid n-superhypergraph. □

Theorem 3.10 (Generalization of Factor Hypergraphs)**.** *Every factor hypergraph $H = (V, E, \{\psi_e\}_{e\in E})$ (Definition 3.3) embeds as a factor 1-superhypergraph $\mathcal{F}^{(1)} = (V_1, E_1, \{\psi'_{e'}\})$ by setting*

$$V_1 = \big\{\{v\} \mid v \in V\big\} \subseteq \mathcal{P}(V), \quad E_1 = \{\, e \subseteq V_1 \mid e' = \{\, v \in V : e' \in E\}\},$$

and defining for each original hyperedge $e \subseteq V$

$$\psi'_{\{\,\{v\}:v\in e\}}(x_{\{v\}})_{\{v\}\in e} \;=\; \psi_e\big(x_v\big)_{v\in e}.$$

Under this mapping, the global functions coincide and $\mathcal{F}^{(1)}$ reproduces H.

Proof. Given $H = (V, E, \{\psi_e\})$, each hyperedge $e \in E$ is a nonempty $e \subseteq V$. Replacing every $v \in V$ by the singleton $\{v\}$ yields $V_1 \subseteq \mathcal{P}(V)$ and each $\{\,\{v\} : v \in e\} \subseteq V_1$, so $E_1 \subseteq \mathcal{P}(V_1)$. Defining $\psi'_{\{\,\{v\}:v\in e\}}(x_{\{v\}}) = \psi_e(x_v)$ ensures that $\prod_{e\in E} \psi_e(x_e) = \prod_{e'\in E_1} \psi'_{e'}(x_{e'})$, so the factorization is preserved. Hence H is realized as a factor 1-superhypergraph, showing that factor n-superhypergraphs generalize ordinary factor hypergraphs. □

4 Signal Flow Graphs and Their Extensions

Signal Flow Graphs represent variables as nodes and directed edges with gains showing signal dependencies, used for transfer function analysis. This framework is extended using Hypergraphs and SuperHypergraphs to model more complex and hierarchical activity dependencies.

Definition 4.1 (Signal Flow Graph)**.** (cf. [21–24]) A *signal flow graph* is a triple

$$G = (V, E, \gamma),$$

where

- V is a finite set of *nodes*,
- $E \subseteq V \times V$ is a set of *directed branches*,
- $\gamma : E \to \mathbb{R}$ assigns a *gain* to each branch.

We interpret each branch $(u, v) \in E$ with gain $\gamma(u, v)$ as contributing $\gamma(u, v)\, x_u$ to the signal at node v. Hence the node equations are

$$x_v \;=\; \sum_{(u,v)\in E} \gamma(u, v)\, x_u \;+\; \eta_v, \quad v \in V,$$

where η_v denotes any external input at node v.

Example 4.2 (Feedback Amplifier Circuit as a Signal Flow Graph)**.** Consider a simple feedback amplifier with three signal nodes:

$$V = \{v_{\text{in}},\ v_{\text{amp}},\ v_{\text{out}}\},$$

and directed branches with gains:

$$E = \big\{(v_{\text{in}}, v_{\text{amp}}),\ (v_{\text{amp}}, v_{\text{out}}),\ (v_{\text{out}}, v_{\text{amp}})\big\},$$

$$\gamma(v_{\text{in}}, v_{\text{amp}}) = 1, \quad \gamma(v_{\text{amp}}, v_{\text{out}}) = G, \quad \gamma(v_{\text{out}}, v_{\text{amp}}) = -\beta.$$

Here

- v_{in} is the input voltage node,
- v_{amp} is the amplifier internal node,
- v_{out} is the output voltage node,
- G is the amplifier forward gain,
- β is the feedback factor.

The node equations

$$x_{v_{\text{amp}}} = x_{v_{\text{in}}} - \beta\, x_{v_{\text{out}}}, \quad x_{v_{\text{out}}} = G\, x_{v_{\text{amp}}}$$

recover the standard feedback relation.

Definition 4.3 (Signal Flow Hypergraph)**.** A *signal flow hypergraph* is a quintuple

$$H = (V, E, t, h, \gamma),$$

where

- (V, E, t, h) is a directed hypergraph with $t : E \to \mathcal{P}(V) \setminus \{\emptyset\}$, $h : E \to \mathcal{P}(V)$ and $|h(e)| = 1$ for all $e \in E$,
- $\gamma : E \to \mathbb{R}$ assigns a *gain* to each hyperedge.

Each hyperedge e with $t(e) = T$ and $h(e) = \{v\}$ contributes $\gamma(e) \sum_{u \in T} x_u$ to node v, so the node equations become

$$x_v \;=\; \sum_{\substack{e \in E \\ h(e) = \{v\}}} \gamma(e) \sum_{u \in t(e)} x_u \;+\; \eta_v, \quad v \in V.$$

Example 4.4 (Sensor Fusion as a Signal Flow Hypergraph)**.** Model a three-sensor fusion node that combines temperature readings:

$$V = \{S_1, S_2, S_3, F\},$$

and hyperedges

$$E = \{e\}, \quad t(e) = \{S_1, S_2, S_3\}, \quad h(e) = \{F\}, \quad \gamma(e) = \tfrac{1}{3}.$$

Here each sensor node S_i provides x_{S_i}, and the fused output at F is

$$x_F = \tfrac{1}{3}\big(x_{S_1} + x_{S_2} + x_{S_3}\big).$$

This single-head hyperedge encodes equal-weight averaging of three inputs.

Theorem 4.5 (Signal Flow Hypergraphs are Hypergraphs)**.** *Let $H = (V, E, t, h, \gamma)$ be a signal flow hypergraph. Then the pair*

$$\big(V,\ E_H\big), \quad E_H = \{\, t(e) \cup h(e) \mid e \in E\},$$

is an (undirected) hypergraph.

Proof. For each $e \in E$, $t(e) \neq \emptyset$ and $h(e) \neq \emptyset$, so $t(e) \cup h(e) \subseteq V$ is nonempty. Hence $E_H \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$, which exactly matches the definition of an undirected hypergraph. □

Theorem 4.6 (Generalization of Signal Flow Graphs)**.** *Every signal flow graph $G = (V, E_G, \gamma_G)$ embeds naturally as a signal flow hypergraph $H = (V, E, t, h, \gamma)$ by setting*

$$E = E_G, \quad t(u, v) = \{u\}, \quad h(u, v) = \{v\}, \quad \gamma(u, v) = \gamma_G(u, v) \quad \textit{for each } (u, v) \in E_G.$$

Proof. Each branch $(u, v) \in E_G$ yields a hyperedge e with nonempty tail $t(e) = \{u\}$ and singleton head $h(e) = \{v\}$, and carries the same gain $\gamma(u, v)$. Thus H satisfies Definition 4.3 and its node equations coincide with those of G, showing that every signal flow graph is a special case of a signal flow hypergraph. □

Definition 4.7 (Signal Flow n-SuperHypergraph)**.** Let V_0 be a finite set of *signal nodes*, each carrying a real-valued signal x_v. For $k \geq 0$, define

$$\mathcal{P}^0(V_0) = V_0, \qquad \mathcal{P}^{k+1}(V_0) = \mathcal{P}\big(\mathcal{P}^k(V_0)\big).$$

A *signal flow n-superhypergraph* is a quintuple

$$\mathrm{SFSH}^{(n)} = \big(V_n,\ E_n,\ t,\ h,\ \gamma\big),$$

where

- $V_n \subseteq \mathcal{P}^n(V_0)$ is the set of n-supervertices,
- $E_n \subseteq \mathcal{P}(V_n) \setminus \{\emptyset\}$ is the set of n-superedges,
- $t : E_n \to \mathcal{P}(V_n) \setminus \{\emptyset\}$ assigns each superedge a nonempty *tail* $t(e) \subseteq V_n$,
- $h : E_n \to \mathcal{P}(V_n)$ assigns each superedge a *head* $h(e) \subseteq V_n$ with $|h(e)| = 1$,
- $\gamma : E_n \to \mathbb{R}$ assigns a real-valued *gain* to each superedge.

The signal at each supervertex $v \in V_n$ satisfies

$$x_v \ = \sum_{\substack{e \in E_n \\ h(e) = \{v\}}} \gamma(e) \sum_{u \in t(e)} x_u \ + \ \eta_v,$$

where η_v is any external input at v.

Example 4.8 (Hierarchical Signal Processing as a Signal Flow 2-SuperHypergraph)**.** In a multi-stage processing chain, group sensor fusion and control stages:

$$V_0 = \{S_1, S_2, S_3, C\}.$$

Form 1-supervertices (processing modules):

$$v_A = \{S_1, S_2, S_3\}, \quad v_B = \{C\}, \quad V_1 = \{v_A, v_B\}.$$

At level 2, wrap into a system-level supervertex:

$$v_{\mathrm{sys}} = \{v_A, v_B\}, \quad V_2 = \{\, \{v_A\},\ \{v_B\},\ v_{\mathrm{sys}} \,\}.$$

Define superedges in E_2 by

$$e_1 :\ t(e_1) = \{\{v_A\}\},\ h(e_1) = \{\{v_B\}\},\ \gamma(e_1) = 1,$$

$$e_2 :\ t(e_2) = \{\{v_A\}, \{v_B\}\},\ h(e_2) = \{v_{\mathrm{sys}}\},\ \gamma(e_2) = 1.$$

Here

- e_1 models the control module v_B receiving fused data from v_A,
- e_2 models the overall system output at supervertex v_{sys} depending on both v_A and v_B.

Thus $\mathrm{SFSH}^{(2)} = (V_2, \{e_1, e_2\}, t, h, \gamma)$ is a valid signal flow 2-superhypergraph encoding a two-layer processing pipeline.

Theorem 4.9 (Signal Flow n-SuperHypergraphs Are n-SuperHyperGraphs)**.** *If* $\mathrm{SFSH}^{(n)} = (V_n, E_n, t, h, \gamma)$ *is a signal flow n-superhypergraph, then* (V_n, E_n) *is an n-superhypergraph in the sense of Definition 2.5.*

Proof. By construction, $V_n \subseteq \mathcal{P}^n(V_0)$ and $E_n \subseteq \mathcal{P}(V_n) \setminus \{\emptyset\} = \mathcal{P}^{n+1}(V_0)$. The maps t and h satisfy the single–head and nonempty–tail conditions of the Definition(i), and acyclicity of the induced relation on V_n follows from the absence of feedback loops in the signal equations. Hence (V_n, E_n) fulfills all requirements of an n-superhypergraph. □

Theorem 4.10 (Generalization of Signal Flow Hypergraphs)**.** *Every signal flow hypergraph $H = (V, E, t, h, \gamma)$ (Definition 4.3) embeds naturally as a signal flow* 1*-superhypergraph* $\mathrm{SFSH}^{(1)} = (V_1, E_1, t', h', \gamma')$ *by setting*

$$V_1 = \big\{\{v\} \mid v \in V\big\} \subseteq \mathcal{P}(V), \quad E_1 = E,$$

and defining for each $e \in E$:

$$t'(e) = \{\,\{u\} \mid u \in t(e)\}, \quad h'(e) = \{\,\{v\} \mid v \in h(e)\}, \quad \gamma'(e) = \gamma(e).$$

Proof. Since each head $h(e)$ in H is a singleton and each tail $t(e)$ is nonempty, the transformed maps t' and h' satisfy the signal flow 1-superhypergraph conditions (Definition 4.7). The signal equations are preserved because

$$\sum_{u \in t(e)} x_u = \sum_{\{u\} \in t'(e)} x_{\{u\}}, \quad h'(e) = \{\{v\}\} \implies x_{\{v\}} = x_v.$$

Thus H is realized exactly as $\mathrm{SFSH}^{(1)}$, showing that signal flow n-superhypergraphs generalize signal flow hypergraphs. □

5 Control Flow Graphs and Their Extensions

Control Flow Graphs depict program execution with basic blocks as nodes connected by directed edges indicating possible control transfers (cf. [25–27]). This framework is extended using Hypergraphs and Super-Hypergraphs to model more complex and hierarchical activity dependencies.

Definition 5.1 (Control Flow Graph)**.** (cf. [25–27]) A *control flow graph* is a tuple

$$G = (N,\ E,\ \mathit{entry},\ \mathit{exit}),$$

where

- N is a finite set of *nodes* (e.g. statements or basic blocks),
- $E \subseteq N \times N$ is a set of directed *edges* indicating possible transfers of control,
- $\mathit{entry} \in N$ is the unique *entry* node with no predecessors,
- $\mathit{exit} \in N$ is the unique *exit* node with no successors,
- every node is reachable from *entry* and can reach *exit*.

Example 5.2 (If–Else Statement as a Control Flow Graph)**.** Consider the following pseudo-code:

```
entry:
  if (x > 0) then
    y := 1
  else
    y := -1
  end if
exit
```

We model this as a CFG $G = (N, E, \mathit{entry}, \mathit{exit})$ with

$$N = \{\mathit{entry},\ v_{\text{cond}},\ v_{\text{then}},\ v_{\text{else}},\ v_{\text{merge}},\ \mathit{exit}\},$$

$$E = \{(\mathit{entry}, v_{\text{cond}}),\ (v_{\text{cond}}, v_{\text{then}}),\ (v_{\text{cond}}, v_{\text{else}}),\ (v_{\text{then}}, v_{\text{merge}}),\ (v_{\text{else}}, v_{\text{merge}}),\ (v_{\text{merge}}, \mathit{exit})\}.$$

Here *entry* has no predecessors, *exit* no successors, and every node lies on a path from *entry* to *exit*.

Definition 5.3 (Control Flow Hypergraph)**.** A *control flow hypergraph* is a directed hypergraph

$$H = (N,\ \mathcal{E},\ t,\ h),$$

where

- N is a finite set of *nodes*,
- $\mathcal{E}$ is a finite set of *hyperedges*,
- $t\colon \mathcal{E} \to \mathcal{P}(N) \setminus \{\emptyset\}$ assigns each $e \in \mathcal{E}$ a nonempty *tail* $t(e) \subseteq N$,
- $h\colon \mathcal{E} \to \mathcal{P}(N) \setminus \{\emptyset\}$ assigns each $e \in \mathcal{E}$ a nonempty *head* $h(e) \subseteq N$,

subject to the usual reachability conditions of control flow (every node reachable from an entry hypervertex and able to reach an exit hypervertex).

Example 5.4 (Switch–Case as a Control Flow Hypergraph)**.** Consider a switch–case construct:

```
switch (c):
  case A: ...; break;
  case B: ...; break;
  default: ...;
endswitch
```

Let

$$N = \{switch,\ v_A,\ v_B,\ v_{\text{def}},\ v_{\text{merge}}\},$$

and define hyperedges

$$e_1 :\ t(e_1) = \{switch\},\ h(e_1) = \{v_A, v_B, v_{\text{def}}\},$$
$$e_2 :\ t(e_2) = \{v_A, v_B, v_{\text{def}}\},\ h(e_2) = \{v_{\text{merge}}\}.$$

Then $H = (N, \{e_1, e_2\}, t, h)$ satisfies Definition 5.3, encoding the multi-way branch and subsequent join.

Theorem 5.5 (Hypergraph Structure)**.** *Let $H = (N, \mathcal{E}, t, h)$ be a control flow hypergraph. Then*

$$\big(N,\ \{\, t(e) \cup h(e) \mid e \in \mathcal{E}\}\big)$$

is an (undirected) hypergraph, since each $t(e) \cup h(e)$ is a nonempty subset of N.

Proof. By Definition 5.3, for every $e \in \mathcal{E}$ both $t(e)$ and $h(e)$ are nonempty subsets of N. Hence

$$t(e)\ \cup\ h(e)\ \neq\ \emptyset, \quad t(e) \cup h(e) \subseteq N,$$

so $\{\, t(e) \cup h(e) \mid e \in \mathcal{E}\} \subseteq \mathcal{P}(N) \setminus \{\emptyset\}$, which exactly matches the definition of an undirected hypergraph. □

Theorem 5.6 (Generalization of Control Flow Graphs)**.** *Every control flow graph $G = (N, E, entry, exit)$ can be viewed as a control flow hypergraph*

$$H = (N,\ \mathcal{E},\ t,\ h),$$

by letting

$$\mathcal{E} = \{\, e_{u,v} \mid (u, v) \in E\}, \quad t(e_{u,v}) = \{u\}, \quad h(e_{u,v}) = \{v\},$$

for each $(u, v) \in E$, and preserving the same entry, exit nodes.

Proof. Since $E \subseteq N \times N$, each $(u, v) \in E$ yields a hyperedge $e_{u,v}$ with tail $\{u\}$ and head $\{v\}$, both nonempty. The reachability and acyclicity (modulo loops) properties of G carry over directly to H via the bijection $(u, v) \mapsto e_{u,v}$. Thus H satisfies Definition 5.3 and faithfully generalizes G. □

Definition 5.7 (Control Flow n-SuperHypergraph)**.** Let V_0 be a finite set of *program points* (e.g. basic blocks). For each integer $k \geq 0$, define the iterated powersets

$$\mathcal{P}^0(V_0) = V_0, \quad \mathcal{P}^{k+1}(V_0) = \mathcal{P}\big(\mathcal{P}^k(V_0)\big).$$

A *control flow n-superhypergraph* is a quadruple

$$\mathrm{CFGSH}^{(n)} = \big(V_n,\ E_n,\ t,\ h\big),$$

where

- $V_n \subseteq \mathcal{P}^n(V_0)$ is the set of n-supervertices,
- $E_n \subseteq \mathcal{P}(V_n) \setminus \{\emptyset\}$ is the set of n-superedges,
- $t\colon E_n \to \mathcal{P}(V_n) \setminus \{\emptyset\}$ assigns to each $e \in E_n$ its nonempty *tail* $t(e) \subseteq V_n$,
- $h\colon E_n \to \mathcal{P}(V_n) \setminus \{\emptyset\}$ assigns to each $e \in E_n$ its nonempty *head* $h(e) \subseteq V_n$.

Intuitively, each superedge e models a transfer of control from all supervertices in $t(e)$ to all supervertices in $h(e)$. One may designate particular supervertices in V_n as the *entry* and *exit* points at level n, requiring reachability from entry to each vertex and from each vertex to exit.

Example 5.8 (Nested Function Calls as a Control Flow 2-SuperHypergraph)**.** Model two functions `foo` and `bar`, each with entry/body/exit points:

$$V_0 = \{e\_foo, b\_foo, x\_foo,\ e\_bar, b\_bar, x\_bar\}.$$

First-level supervertices (functions):

$$F_{\text{foo}} = \{e\_foo, b\_foo, x\_foo\}, \quad F_{\text{bar}} = \{e\_bar, b\_bar, x\_bar\},$$
$$V_1 = \{F_{\text{foo}}, F_{\text{bar}}\}.$$

Wrap into second-level supervertices:

$$U_{\text{foo}} = \{F_{\text{foo}}\}, \quad U_{\text{bar}} = \{F_{\text{bar}}\}, \quad U_{\text{prog}} = \{F_{\text{foo}}, F_{\text{bar}}\},$$
$$V_2 = \{\, U_{\text{foo}},\ U_{\text{bar}},\ U_{\text{prog}}\,\}.$$

Define 2-superedges

$$e_1 :\ t(e_1) = \{U_{\text{foo}}\},\ h(e_1) = \{U_{\text{bar}}\},$$
$$e_2 :\ t(e_2) = \{U_{\text{foo}}, U_{\text{bar}}\},\ h(e_2) = \{U_{\text{prog}}\}.$$

Here e_1 models `foo` calling `bar`, and e_2 the overall program completion after both functions finish. Hence $\mathrm{CFGSH}^{(2)} = (V_2, \{e_1, e_2\}, t, h)$ is a valid Control Flow 2-SuperHyperGraph.

Theorem 5.9 (Control Flow n-SuperHypergraphs Are n-SuperHyperGraphs)**.** *If* $\mathrm{CFGSH}^{(n)} = (V_n, E_n, t, h)$ *is a control flow n-superhypergraph as in Definition 5.7, then the pair* (V_n, E_n) *satisfies the conditions of an n-superhypergraph (Definition 2.5).*

Proof. By hypothesis $V_n \subseteq \mathcal{P}^n(V_0)$ and $E_n \subseteq \mathcal{P}(V_n) \setminus \{\emptyset\} = \mathcal{P}^{n+1}(V_0)$. Thus (V_n, E_n) meets precisely the requirement that both supervertices and superedges lie in the nth and $(n+1)$th iterated powersets of V_0. No further conditions are needed for (V_n, E_n) to be an n-superhypergraph. □

Theorem 5.10 (Generalization of Control Flow Hypergraphs)**.** *Every control flow hypergraph* $H = (V, \mathcal{E}, t_H, h_H)$ *(Definition 5.3) embeds naturally as a control flow* 1*-superhypergraph*

$$\mathrm{CFGSH}^{(1)} = (V_1, E_1, t, h),$$

by setting

$$V_1 = \{\{v\} \mid v \in V\} \subseteq \mathcal{P}(V), \quad E_1 = \mathcal{E},$$

and defining

$$t(e) = \{\{u\} \mid u \in t_H(e)\}, \quad h(e) = \{\{v\} \mid v \in h_H(e)\}, \quad \textit{for each } e \in E_1.$$

Proof. Since each $e \in \mathcal{E}$ satisfies $t_H(e) \neq \emptyset$ and $h_H(e) \neq \emptyset$, the transformed tails $t(e)$ and heads $h(e)$ are also nonempty subsets of V_1. Moreover, $V_1 \subseteq \mathcal{P}(V)$ and $E_1 \subseteq \mathcal{P}(V_1) \setminus \{\emptyset\}$, so (V_1, E_1) is a control flow 1-superhypergraph. The reachability and structural properties of H carry over identically, demonstrating that control flow hypergraphs are special cases of control flow n-superhypergraphs. □

6 Data Flow Graphs and Their Extensions

Dataflow graphs model computational actors as nodes and channels as edges, capturing data dependencies for efficient parallel execution and scheduling (cf. [28–31]). This framework is extended using Hypergraphs and SuperHypergraphs to model more complex and hierarchical activity dependencies.

Definition 6.1 (Dataflow Graph)**.** (cf. [28–31]) A *dataflow graph* is a tuple

$$D = \big(V,\ E,\ \mathrm{src},\ \mathrm{dst},\ \mathrm{prd},\ \mathrm{cns},\ \gamma_0\big),$$

where

- V is a finite set of *actors*,
- E is a finite set of *channels*,
- $\mathrm{src}, \mathrm{dst} : E \to V$ assign to each channel its unique source and sink actor,
- $\mathrm{prd}, \mathrm{cns} : E \to \mathbb{N}_{>0}$ specify the *production* and *consumption rates*,
- $\gamma_0 : E \to \mathbb{N}$ gives the *initial token count* on each channel.

A firing of an actor $u \in V$ removes $\mathrm{cns}(e)$ tokens from each incoming channel e with $\mathrm{dst}(e) = u$, and adds $\mathrm{prd}(e)$ tokens to each outgoing channel e with $\mathrm{src}(e) = u$.

Example 6.2 (Streaming FIR Filter as a Dataflow Graph)**.** Consider a simple 3-tap finite impulse response (FIR) filter implemented in a streaming DSP pipeline:

$$V = \{\, \mathit{Read},\ \mathit{MAC},\ \mathit{Write} \,\},$$

$$E = \{\, e_1, e_2 \},$$

with

$$\begin{aligned} &\mathrm{src}(e_1) = \mathit{Read}, \quad \mathrm{dst}(e_1) = \mathit{MAC}, \quad \mathrm{prd}(e_1) = 1,\ \mathrm{cns}(e_1) = 1,\ \gamma_0(e_1) = 0, \\ &\mathrm{src}(e_2) = \mathit{MAC}, \quad \mathrm{dst}(e_2) = \mathit{Write}, \quad \mathrm{prd}(e_2) = 1,\ \mathrm{cns}(e_2) = 1,\ \gamma_0(e_2) = 0. \end{aligned}$$

Here:

- *Read* reads one sample per firing,
- *MAC* multiplies the latest three samples by fixed coefficients and accumulates them,
- *Write* outputs each filtered sample.

Firing *MAC* removes one token from channel e_1 and produces one token on e_2, modeling the streaming filter behavior.

Definition 6.3 (Dataflow Hypergraph)**.** A *directed dataflow hypergraph* is a tuple

$$H = \big(V,\ E_H,\ t,\ h,\ \mathrm{prd}_H,\ \mathrm{cns}_H,\ \gamma_0^H\big),$$

where

- V is a finite set of *actors*,
- E_H is a finite set of *hyperchannels*,
- $t, h : E_H \to \mathcal{P}(V) \setminus \{\emptyset\}$ assign to each hyperchannel nonempty sets of *producers* $t(e)$ and *consumers* $h(e)$,
- $\mathrm{prd}_H : E_H \times V \to \mathbb{N}_{>0}$ gives the production rate on each tail actor ($\mathrm{prd}_H(e, u) > 0$ only if $u \in t(e)$),

- $\mathrm{cns}_H : E_H \times V \to \mathbb{N}_{>0}$ gives the consumption rate on each head actor ($\mathrm{cns}_H(e, v) > 0$ only if $v \in h(e)$),
- $\gamma_0^H : E_H \to \mathbb{N}$ is the initial token count for each hyperchannel.

A firing of a hyperchannel $e \in E_H$ removes $\mathrm{cns}_H(e, v)$ tokens from each $v \in h(e)$ and adds $\mathrm{prd}_H(e, u)$ tokens to each $u \in t(e)$.

Example 6.4 (Sensor Fusion as a Dataflow Hypergraph)**.** To fuse readings from three temperature sensors, we use a directed dataflow hypergraph:

$$V = \{S_1, S_2, S_3, \mathit{Fuse}\},$$
$$E_H = \{\, e\},$$

with

$$t(e) = \{S_1, S_2, S_3\}, \quad h(e) = \{\mathit{Fuse}\},$$
$$\mathrm{prd}_H(e, S_i) = 1, \quad \mathrm{cns}_H(e, \mathit{Fuse}) = 3, \quad \gamma_0^H(e) = 0.$$

Here the hyperchannel e collects one token from each sensor S_1, S_2, S_3 (consuming three tokens total) and produces one fused token at *Fuse*. This models equal-weight averaging of three inputs in a single step.

Theorem 6.5 (Underlying Hypergraph)**.** *Let $H = (V, E_H, t, h, \dots)$ be a dataflow hypergraph (Definition 6.3). Then*

$$\big(V,\ E'\big), \quad E' = \{\, t(e) \cup h(e) \mid e \in E_H\},$$

is an (undirected) hypergraph, since each $t(e) \cup h(e) \subseteq V$ is nonempty.

Proof. By Definition 6.3, for every $e \in E_H$ we have $t(e) \neq \emptyset$ and $h(e) \neq \emptyset$. Hence $t(e) \cup h(e) \subseteq V$ is a nonempty subset, so $E' \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$, satisfying the hypergraph definition. □

Theorem 6.6 (Generalization of Dataflow Graphs)**.** *Every dataflow graph*

$$D = \big(V, E, \mathrm{src}, \mathrm{dst}, \mathrm{prd}, \mathrm{cns}, \gamma_0\big)$$

induces a dataflow hypergraph

$$H = \big(V, E_H, t, h, \mathrm{prd}_H, \mathrm{cns}_H, \gamma_0^H\big)$$

by setting

$$E_H = \{\, e' \mid e \in E\}, \quad t(e') = \{\mathrm{src}(e)\}, \quad h(e') = \{\mathrm{dst}(e)\},$$
$$\mathrm{prd}_H(e', u) = \begin{cases} \mathrm{prd}(e), & u = \mathrm{src}(e), \\ 0, & \mathit{otherwise}, \end{cases} \qquad \mathrm{cns}_H(e', v) = \begin{cases} \mathrm{cns}(e), & v = \mathrm{dst}(e), \\ 0, & \mathit{otherwise}, \end{cases}$$
$$\gamma_0^H(e') = \gamma_0(e).$$

Proof. Each $e' \in E_H$ has a singleton tail $\{\mathrm{src}(e)\}$ and head $\{\mathrm{dst}(e)\}$, so $t(e')$ and $h(e')$ are nonempty. The rate functions and initial tokens agree by construction. Therefore H meets all requirements of Definition 6.3, and it clearly reduces to D when each hyperchannel is viewed as an ordinary channel. Hence dataflow hypergraphs strictly generalize dataflow graphs. □

Definition 6.7 (Dataflow n-SuperHypergraph)**.** Let V_0 be a finite set of *actors*. For each integer $k \geq 0$, define

$$\mathcal{P}^0(V_0) = V_0, \quad \mathcal{P}^{k+1}(V_0) = \mathcal{P}\big(\mathcal{P}^k(V_0)\big).$$

A *dataflow n-superhypergraph* is a tuple

$$\mathcal{D}^{(n)} = \big(V_n,\ E_n,\ t,\ h,\ \mathrm{prd}^{(n)},\ \mathrm{cns}^{(n)},\ \gamma_0^{(n)}\big),$$

where

- $V_n \subseteq \mathcal{P}^n(V_0)$ is the set of n-supervertices,
- $E_n \subseteq \mathcal{P}^n(V_0)$ is the set of n-superedges,

- $t, h : E_n \to \mathcal{P}(V_n)$ assign to each $e \in E_n$ a nonempty *tail* $t(e) \subseteq V_n$ and a nonempty *head* $h(e) \subseteq V_n$,
- $\mathrm{prd}^{(n)} : E_n \times V_n \to \mathbb{N}_{>0}$ gives the *production rate*, with $\mathrm{prd}^{(n)}(e, u) > 0$ only if $u \in t(e)$,
- $\mathrm{cns}^{(n)} : E_n \times V_n \to \mathbb{N}_{>0}$ gives the *consumption rate*, with $\mathrm{cns}^{(n)}(e, v) > 0$ only if $v \in h(e)$,
- $\gamma_0^{(n)} : E_n \to \mathbb{N}$ specifies the *initial token count* on each n-superedge.

Firing an n-superedge e removes $\mathrm{cns}^{(n)}(e, v)$ tokens from each head supervertex $v \in h(e)$ and adds $\mathrm{prd}^{(n)}(e, u)$ tokens to each tail supervertex $u \in t(e)$.

Example 6.8 (Hierarchical Image Processing as a Dataflow 2-SuperHyperGraph)**.** Consider an image processing pipeline with actors:

$$V_0 = \{\mathit{Load}, \mathit{Filter}, \mathit{Encode}, \mathit{Store}\}.$$

First-level modules (1-supervertices):

$$M_1 = \{\mathit{Load}, \mathit{Filter}\}, \quad M_2 = \{\mathit{Encode}, \mathit{Store}\},$$

$$V_1 = \{M_1,\ M_2\} \subseteq \mathcal{P}(V_0).$$

Second-level system grouping (2-supervertices):

$$S_{\text{in}} = \{M_1\}, \quad S_{\text{out}} = \{M_2\}, \quad S_{\text{all}} = \{M_1, M_2\},$$

$$V_2 = \{S_{\text{in}},\ S_{\text{out}},\ S_{\text{all}}\} \ \subseteq\ \mathcal{P}^2(V_0).$$

Define hyperedges in E_2 by

$$e_1 :\ t(e_1) = \{S_{\text{in}}\},\ h(e_1) = \{S_{\text{out}}\}, \quad e_2 :\ t(e_2) = \{S_{\text{in}}, S_{\text{out}}\},\ h(e_2) = \{S_{\text{all}}\}.$$

Interpretation:

- e_1 models the filter module M_2 depending on data from load/filter module M_1.
- e_2 models the overall system completion (S_{all}) depending on both modules.

Since each head is a singleton and the induced relation on V_2 is acyclic, $\mathcal{D}^{(2)} = (V_2, \{e_1, e_2\}, t, h, \dots)$ is a valid Dataflow 2-SuperHyperGraph.

Theorem 6.9 (Dataflow n-SuperHypergraphs Are n-SuperHyperGraphs)**.** *If $\mathcal{D}^{(n)} = (V_n, E_n, t, h, \mathrm{prd}^{(n)}, \mathrm{cns}^{(n)}, \gamma_0^{(n)})$ is a dataflow n-superhypergraph, then the pair (V_n, E_n) satisfies Definition 2.5 and is therefore an n-superhypergraph.*

Proof. By construction, $V_n \subseteq \mathcal{P}^n(V_0)$ and $E_n \subseteq \mathcal{P}^n(V_0)$. Hence

$$V_n \subseteq \mathcal{P}^n(V_0), \quad E_n \subseteq \mathcal{P}^n(V_0),$$

which matches the requirements of an n-superhypergraph (Definition 2.5). No further conditions are needed. □

Theorem 6.10 (Generalization of Dataflow Hypergraphs)**.** *Every dataflow hypergraph*

$$H = \big(V,\ E_H,\ t_H,\ h_H,\ \mathrm{prd}_H,\ \mathrm{cns}_H,\ \gamma_0^H\big)$$

(Definition 6.3) embeds as a dataflow 1*-superhypergraph $\mathcal{D}^{(1)} = (V_1, E_1, t, h, \mathrm{prd}^{(1)}, \mathrm{cns}^{(1)}, \gamma_0^{(1)})$ by setting*

$$V_1 = \big\{\{v\} \mid v \in V\big\} \ \subseteq\ \mathcal{P}(V_0), \quad E_1 = E_H,$$

and, for each $e \in E_1$ and each $\{u\} \in V_1$,

$$t(e) \ = \ \big\{\{u\} \mid u \in t_H(e)\big\}, \quad h(e) \ = \ \big\{\{v\} \mid v \in h_H(e)\big\},$$

$$\mathrm{prd}^{(1)}(e, \{u\}) = \begin{cases} \mathrm{prd}_H(e, u), & u \in t_H(e), \\ 0, & \textit{otherwise}, \end{cases} \qquad \mathrm{cns}^{(1)}(e, \{v\}) = \begin{cases} \mathrm{cns}_H(e, v), & v \in h_H(e), \\ 0, & \textit{otherwise}, \end{cases}$$

$$\gamma_0^{(1)}(e) = \gamma_0^H(e).$$

Proof. Since each $t_H(e)$ and $h_H(e)$ is nonempty, the transformed tails $t(e)$ and heads $h(e)$ are nonempty subsets of V_1. Moreover, $\{v\} \in V_1$ implies $V_1 \subseteq \mathcal{P}(V_0) = \mathcal{P}^1(V_0)$ and $E_1 = E_H \subseteq \mathcal{P}(V) \subseteq \mathcal{P}(V_1)$. Thus $\mathcal{D}^{(1)}$ meets all conditions of Definition 6.7 for $n = 1$, and its production, consumption, and initial token functions agree with those of H. Therefore dataflow hypergraphs are faithfully realized as dataflow 1-superhypergraphs, proving the claimed generalization. □

7 PERT/CPM Graph and Their Extensions

A PERT/CPM Graph is a directed acyclic graph representing project activities as edges and events as vertices to calculate schedules. This framework is extended using Hypergraphs and SuperHypergraphs to model more complex and hierarchical activity dependencies.

Definition 7.1 (PERT/CPM Graph)**.** (cf. [32]) A *PERT/CPM Graph* is a quintuple

$$G = \big(V, E, d, s, t\big),$$

where

- V is a finite set of *events* (vertices), including a *start* event s and a *finish* event t, with $s, t \in V$ and $s \neq t$.
- $E \subseteq V \times V$ is a set of directed edges (activities) satisfying:
  1. $(u, v) \in E \implies u \neq v$ (no loops).
  2. G is acyclic.
  3. s has no predecessors and t has no successors.
- $d : E \to \mathbb{R}_{>0}$ assigns each activity (u, v) its *duration*.

Example 7.2 (House Construction as a PERT/CPM Graph)**.** Model the main stages of building a house:

$$V = \{\, s,\ v_1,\ v_2,\ v_3,\ v_4,\ t\},$$

where

$$s = \text{project start},\ v_1 = \text{foundation complete},\ v_2 = \text{framing complete},$$
$$v_3 = \text{roofing complete},\ v_4 = \text{finishing complete},\ t = \text{move–in ready}.$$

Activities and durations:

$$E = \{(s, v_1), (v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, t)\},$$
$$d(s, v_1) = 7,\ d(v_1, v_2) = 14,\ d(v_2, v_3) = 5,\ d(v_3, v_4) = 10,\ d(v_4, t) = 2.$$

This acyclic graph with start s (no predecessors) and finish t (no successors) encodes the earliest- and latest-time calculations for each event via forward/backward passes.

Definition 7.3 (Forward and Backward Pass)**.** Define the *earliest occurrence time* $\mathrm{EOT} : V \to \mathbb{R}_{\geq 0}$ by

$$\mathrm{EOT}(v) \;=\; \begin{cases} 0, & v = s, \\ \max\limits_{(u,v)\in E} \big(\mathrm{EOT}(u) + d(u, v)\big), & v \neq s. \end{cases}$$

Define the *latest occurrence time* $\mathrm{LOT} : V \to \mathbb{R}_{\geq 0}$ by

$$\mathrm{LOT}(v) \;=\; \begin{cases} \mathrm{EOT}(t), & v = t, \\ \min\limits_{(v,w)\in E} \big(\mathrm{LOT}(w) - d(v, w)\big), & v \neq t. \end{cases}$$

Definition 7.4 (Slack Times)**.** For each activity $(u, v) \in E$, define

$$\text{Total Slack } S_T(u, v) = \mathrm{LOT}(v) - \mathrm{EOT}(u) - d(u, v),$$

$$\text{Free Slack } S_F(u, v) = \min_{(v,w)\in E} \big(\mathrm{EOT}(w)\big) - \mathrm{EOT}(u) - d(u, v).$$

Definition 7.5 (Critical Path)**.** An *activity* $(u, v) \in E$ is *critical* if $S_T(u, v) = 0$. A *critical path* is any directed path from s to t consisting entirely of critical activities. The project *duration* is $\mathrm{EOT}(t)$.

Definition 7.6 (PERT/CPM Hypergraph)**.** A *PERT/CPM hypergraph* is a quintuple

$$\mathcal{H} = \big(V,\ \mathcal{E},\ t,\ h,\ d\big),$$

where

- V is a finite set of *events*, including distinguished start $s \in V$ and finish $t \in V$ with $s \neq t$.
- $\mathcal{E}$ is a finite set of *hyperactivities*.
- $t \colon \mathcal{E} \to \mathcal{P}(V) \setminus \{\emptyset\}$ assigns to each hyperactivity e its (nonempty) set of *predecessor events*.
- $h \colon \mathcal{E} \to \mathcal{P}(V) \setminus \{\emptyset\}$ assigns to each e its (nonempty) set of *successor events*.
- $d \colon \mathcal{E} \to \mathbb{R}_{>0}$ gives the *duration* of each hyperactivity.

We require that the induced reachability relation on V is acyclic, that s has no predecessors and t no successors, and that every $v \in V$ lies on some directed hyperpath from s to t.

Example 7.7 (New Product Launch as a PERT/CPM Hypergraph)**.** Define the key events:

$$V = \{\, e_{\text{design}},\ e_{\text{prototype}},\ e_{\text{testing}},\ e_{\text{marketing}},\ e_{\text{launch}} \}.$$

We capture the fact that the *launch* cannot occur until design, prototype, testing, and marketing are all complete, via a single hyperactivity:

$$\mathcal{E} = \{\, e_{\mathrm{L}} \},$$

$$t(e_{\mathrm{L}}) = \{e_{\text{design}}, e_{\text{prototype}}, e_{\text{testing}}, e_{\text{marketing}}\}, \quad h(e_{\mathrm{L}}) = \{e_{\text{launch}}\}, \quad d(e_{\mathrm{L}}) = 1.$$

Here $d(e_{\mathrm{L}})$ models the final launch ceremony duration. Since the induced reachability is acyclic, $s = e_{\text{design}}$ has no predecessors and $t = e_{\text{launch}}$ no successors, this is a valid PERT/CPM hypergraph.

Theorem 7.8 (Underlying Hypergraph)**.** *Let $\mathcal{H} = (V, \mathcal{E}, t, h, d)$ be a PERT/CPM hypergraph. Then*

$$H = \big(V,\ E_H\big), \quad E_H = \{\, t(e) \cup h(e) \mid e \in \mathcal{E} \}$$

is an undirected hypergraph in the sense of Definition 2.3: each $t(e) \cup h(e)$ is a nonempty subset of V.

Proof. By Definition 7.6, for every $e \in \mathcal{E}$ both $t(e)$ and $h(e)$ are nonempty subsets of V. Hence $t(e) \cup h(e) \subseteq V$ is nonempty, and so $E_H \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$, which exactly matches the standard definition of an undirected hypergraph. □

Theorem 7.9 (Generalization of PERT/CPM Graphs)**.** *Every classical PERT/CPM graph*

$$G = (V, E, d, s, t)$$

embeds naturally as a PERT/CPM hypergraph $\mathcal{H} = (V, \mathcal{E}, t_{\mathcal{H}}, h_{\mathcal{H}}, d_{\mathcal{H}})$ by setting

$$\mathcal{E} = \{\, e_{u,v} \mid (u, v) \in E \}, \quad t_{\mathcal{H}}(e_{u,v}) = \{u\}, \quad h_{\mathcal{H}}(e_{u,v}) = \{v\}, \quad d_{\mathcal{H}}(e_{u,v}) = d(u, v).$$

Proof. Since each edge $(u, v) \in E$ has $u \neq v$, defining $e_{u,v}$ with tail $\{u\}$ and head $\{v\}$ yields a hyperactivity satisfying the PERT/CPM hypergraph conditions. The reachability and acyclicity of G carry over to $\mathcal{H}$, and durations agree by construction. Thus $\mathcal{H}$ is a PERT/CPM hypergraph whose special case of singleton tails and heads reproduces the original PERT/CPM graph. □

Definition 7.10 (PERT/CPM n-SuperHypergraph)**.** Let V_0 be a finite set of *events*, including distinguished start $s \in V_0$ and finish $t \in V_0$ with $s \neq t$. For $k \geq 0$, define iterated powersets

$$\mathcal{P}^0(V_0) = V_0, \quad \mathcal{P}^{k+1}(V_0) = \mathcal{P}(\mathcal{P}^k(V_0)).$$

A *PERT/CPM n-superhypergraph* is a tuple

$$\mathcal{H}^{(n)} = \big(V_n,\ E_n,\ t,\ h,\ d\big),$$

where

- $V_n \subseteq \mathcal{P}^n(V_0)$ is the set of n-supervertices, containing the *start supervertex* s_n and *finish supervertex* t_n,
- $E_n \subseteq \mathcal{P}(V_n) \setminus \{\emptyset\}$ is the set of n-superactivities,
- $t : E_n \to \mathcal{P}(V_n) \setminus \{\emptyset\}$ assigns each $e \in E_n$ its nonempty set of *predecessor supervertices*,
- $h : E_n \to \mathcal{P}(V_n) \setminus \{\emptyset\}$ assigns each $e \in E_n$ its nonempty set of *successor supervertices*,
- $d : E_n \to \mathbb{R}_{>0}$ gives the *duration* of each superactivity.

These must satisfy:

(i) Acyclicity: The reachability relation induced by (t, h) is a partial order on V_n.

(ii) Boundary: s_n has no predecessors and t_n has no successors.

(iii) Connectivity: Every $v \in V_n$ lies on some directed hyperpath from s_n to t_n.

Example 7.11 (Software Project Milestone as a PERT/CPM 2-SuperHypergraph)**.** Let base events be

$$V_0 = \{\ p = \text{planning\_done},\ i = \text{implementation\_done},\ t = \text{testing\_done}\}.$$

First-level supervertices (phases):

$$P = \{p\}, \quad I = \{i\}, \quad T = \{t\}, \quad V_1 = \{P, I, T\} \subseteq \mathcal{P}(V_0).$$

Second-level supervertices (milestones):

$$M_1 = \{P, I\}, \quad M_2 = \{P, I, T\}, \quad V_2 = \{M_1, \{T\}, M_2\} \subseteq \mathcal{P}^2(V_0).$$

Define superactivities in $E_2 \subseteq \mathcal{P}(V_1)$:

$$e_1 :\ t(e_1) = \{M_1\},\ h(e_1) = \{\{T\}\},\ d(e_1) = 5,$$

$$e_2 :\ t(e_2) = \{M_1, \{T\}\},\ h(e_2) = \{M_2\},\ d(e_2) = 2.$$

Here e_1 encodes that testing (T) depends on planning+implementation (M_1), and e_2 that the final milestone (M_2) depends on both phases. No directed cycles occur, so $\mathcal{H}^{(2)} = (V_2, E_2, t, h, d)$ is a valid PERT/CPM 2-superhypergraph.

Theorem 7.12 (PERT/CPM n-SuperHypergraphs Are n-SuperHyperGraphs)**.** *If $\mathcal{H}^{(n)} = (V_n, E_n, t, h, d)$ is a PERT/CPM n-superhypergraph, then the pair (V_n, E_n) satisfies the definition of an n-superhypergraph (Definition 2.5).*

Proof. By construction $V_n \subseteq \mathcal{P}^n(V_0)$ and

$$E_n \subseteq \mathcal{P}(V_n) = \mathcal{P}^{n+1}(V_0).$$

Thus (V_n, E_n) meets exactly the requirement that supervertices lie in $\mathcal{P}^n(V_0)$ and superedges in $\mathcal{P}^{n+1}(V_0)$. No further conditions are needed for (V_n, E_n) to be an n-superhypergraph. □

Theorem 7.13 (Generalization of PERT/CPM Hypergraphs)**.** *Every PERT/CPM hypergraph* $H = (V_0, \mathcal{E}, t_H, h_H, d_H)$ *(Definition 7.6) embeds as a PERT/CPM* 1*-superhypergraph* $\mathcal{H}^{(1)} = (V_1, E_1, t, h, d)$ *by setting*

$$V_1 = \{\{v\} \mid v \in V_0\} \subseteq \mathcal{P}(V_0), \quad E_1 = \mathcal{E},$$

and for each $e \in E_1$*:*

$$t(e) = \{\{u\} \mid u \in t_H(e)\}, \quad h(e) = \{\{v\} \mid v \in h_H(e)\}, \quad d(e) = d_H(e).$$

Moreover, the start and finish supervertices are $s_1 = \{s\}$ *and* $t_1 = \{t\}$*.*

Proof. Since each hyperactivity $e \in \mathcal{E}$ has nonempty predecessors $t_H(e)$ and successors $h_H(e)$, the mapped sets $t(e)$ and $h(e)$ are nonempty subsets of V_1. Clearly $V_1 \subseteq \mathcal{P}(V_0)$ and $E_1 \subseteq \mathcal{P}(V_1)$. Acyclicity, boundary, and connectivity conditions follow directly because hyperpaths in $\mathcal{H}^{(1)}$ correspond bijectively to those in H. Hence $\mathcal{H}^{(1)}$ is a valid PERT/CPM 1-superhypergraph, showing that PERT/CPM n-superhypergraphs generalize PERT/CPM hypergraphs. □

8 Conclusion and Future Work

We have shown that five foundational graph models—factor graphs, signal flow graphs, control flow graphs, dataflow graphs, and PERT/CPM graphs—can all be embedded within the unified frameworks of Hypergraphs and SuperHypergraphs, and we provided concrete examples illustrating each extension.

For future research, we plan to incorporate uncertainty-aware extensions by leveraging Fuzzy Sets [50], HyperFuzzy Sets [51–53], Intuitionistic Fuzzy Sets [39, 54], Picture Fuzzy Sets [55], Hesitant Fuzzy Sets [56], Neutrosophic Sets [57–59], QuadriPartitioned Neutrosophic Sets [60], and Plithogenic Sets [61, 62]. These enhancements promise richer, more flexible modeling of dependencies in environments characterized by ambiguity or multi-valued information.

Funding

The authors received no grants, contracts, or in-kind contributions for this study.

Acknowledgements

We are indebted to the colleagues, reviewers, and mentors whose suggestions sharpened our arguments. We also recognise the authors cited throughout the paper for providing the intellectual groundwork on which this study builds. Finally, we thank the institutions and individuals who supplied the computing resources and working environment that made the writing of this manuscript possible.

Data Availability

The work is conceptual and does not rely on empirical data; consequently, no datasets were generated or analysed. Researchers interested in empirical validation are encouraged to design their own studies using the theoretical framework presented here.

Ethical Approval

Because the research involves no human or animal subjects, formal ethical approval is unnecessary.

Conflicts of Interest

The authors declare that they have no competing financial or non-financial interests.

Disclaimer

The results and views expressed are those of the authors alone. Although care has been taken to ensure accuracy and proper citation, inadvertent errors may remain. Readers should verify any critical information independently, and practitioners should test the proposed ideas in real-world contexts before implementation.

References

[1] Reinhard Diestel. *Graph theory*. Springer (print edition); Reinhard Diestel (eBooks), 2024.

[2] Jonathan L Gross, Jay Yellen, and Mark Anderson. *Graph theory and its applications*. Chapman and Hall/CRC, 2018.

[3] Claude Berge. *Hypergraphs: combinatorics of finite sets*, volume 45. Elsevier, 1984.

[4] Florentin Smarandache. *Introduction to the n-SuperHyperGraph-the most general form of graph today*. Infinite Study, 2022.

[5] Mohammad Hamidi, Florentin Smarandache, and Mohadeseh Taghinezhad. *Decision Making Based on Valued Fuzzy Superhypergraphs*. Infinite Study, 2023.

[6] Reinhard Diestel. Graph theory 3rd ed. *Graduate texts in mathematics*, 173(33):12, 2005.

[7] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3558–3565, 2019.

[8] Derun Cai, Moxian Song, Chenxi Sun, Baofeng Zhang, Shenda Hong, and Hongyan Li. Hypergraph structure learning for hypergraph neural networks. In *IJCAI*, pages 1923–1929, 2022.

[9] Bibin K Jose and Zsolt Tuza. Hypergraph domination and strong independence. *Applicable Analysis and Discrete Mathematics*, 3(2):347–358, 2009.

[10] Takaaki Fujita and Florentin Smarandache. Fundamental computational problems and algorithms for superhypergraphs. *HyperSoft Set Methods in Engineering*, 3:32–61, 2025.

[11] N. B. Nalawade, M. S. Bapat, S. G. Jakkewad, G. A. Dhanorkar, and D. J. Bhosale. Structural properties of zero-divisor hypergraph and superhypergraph over $\mathbb{Z}_n$: Girth and helly property. *Panamerican Mathematical Journal*, 35(4S):485, 2025.

[12] Florentin Smarandache. *Introduction to the n-SuperHyperGraph-the most general form of graph today*. Infinite Study, 2022.

[13] Mohammad Hamidi, Florentin Smarandache, and Elham Davneshvar. Spectrum of superhypergraphs via flows. *Journal of Mathematics*, 2022(1):9158912, 2022.

[14] Takaaki Fujita and Florentin Smarandache. A concise study of some superhypergraph classes. *Neutrosophic Sets and Systems*, 77:548–593, 2024.

[15] Mohammad Hamidi and Mohadeseh Taghinezhad. *Application of Superhypergraphs-Based Domination Number in Real World*. Infinite Study, 2023.

[16] Idan Schwartz, Seunghak Yu, Tamir Hazan, and Alexander G Schwing. Factor graph attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2039–2048, 2019.

[17] Frank Dellaert. Factor graphs: Exploiting structure in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):141–166, 2021.

[18] Brendan J Frey, Frank R Kschischang, Hans-Andrea Loeliger, and Niclas Wiberg. Factor graphs and algorithms. In *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, volume 35, pages 666–680. Citeseer, 1997.

[19] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. *Georgia Institute of Technology, Tech. Rep*, 2(4), 2012.

[20] Uriel Feige and Shlomo Jozeph. Universal factor graphs. In *Automata, Languages, and Programming: 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I 39*, pages 339–350. Springer, 2012.

[21] Mummadi Veerachary. General rules for signal flow graph modeling and analysis of dc-dc converters. *IEEE transactions on Aerospace and Electronic Systems*, 40(1):259–271, 2004.

[22] William H Huggins. Signal-flow graphs and random signals. *Proceedings of the IRE*, 45(1):74–86, 2007.

[23] Janusz A Brzozowski and Edward J McCluskey. Signal flow graph techniques for sequential circuit state diagrams. *IEEE Transactions on Electronic Computers*, (2):67–76, 2006.

[24] Feim Ridvan Rasim and Sebastian M Sattler. Analysis of electronic circuits with the signal flow graph method. *Circuits and Systems*, 8(11):261–274, 2017.

[25] Henrik Theiling. Control flow graphs for real-time systems analysis. *Universität des Saarlandes, Diss*, 2002.

[26] Sabin Devkota. *Visualizing Control Flow Graphs*. PhD thesis, The University of Arizona, 2021.

[27] Stephan Arlt, Philipp Rümmer, and Martin Schäf. A theory for control-flow graph exploration. In *Automated Technology for Verification and Analysis: 11th International Symposium, ATVA 2013, Hanoi, Vietnam, October 15-18, 2013. Proceedings*, pages 506–515. Springer, 2013.

[28] Shuvra S Bhattacharyya, Ed F Deprettere, and Bart D Theelen. Dynamic dataflow graphs. *Handbook of Signal Processing Systems*, pages 905–944, 2013.

[29] Edward A Lee. Consistency in dataflow graphs. *IEEE Transactions on Parallel and Distributed systems*, 2(2):223–235, 1991.

[30] Klaus Schneider and Anoop Bhagyanath. Consistency constraints for mapping dataflow graphs to hybrid dataflow/von neumann architectures. *ACM Transactions on Embedded Computing Systems*, 22(5):1–25, 2023.

[31] Shuvra S Bhattacharyya, Praveen K Murthy, and Edward A Lee. *Software synthesis from dataflow graphs*, volume 360. Springer Science & Business Media, 2012.

[32] Mete Mazlum and Ali Fuat Güneri. Cpm, pert and project management with fuzzy logic technique and implementation on a business. *Procedia-Social and Behavioral Sciences*, 210:348–357, 2015.

[33] Takaaki Fujita and Prem Kumar Singh. Hyperfuzzy graph and hyperfuzzy hypergraph. *Journal of Neutrosophic and Fuzzy Systems (JNFS)*, 10(01):01–13, 2025.

[34] Anam Luqman, Muhammad Akram, and Ahmad N. Al-Kenani. q-rung orthopair fuzzy hypergraphs with applications. *Mathematics*, 7(3):260, 2019.

[35] Yue Gao, Zizhao Zhang, Haojie Lin, Xibin Zhao, Shaoyi Du, and Changqing Zou. Hypergraph learning: Methods and practices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2548–2566, 2020.

[36] Yuxin Wang, Quan Gan, Xipeng Qiu, Xuanjing Huang, and David Wipf. From hypergraph energy functions to hypergraph neural networks. In *International Conference on Machine Learning*, pages 35605–35623. PMLR, 2023.

[37] Eduardo Martín Campoverde Valencia, Jessica Paola Chuisaca Vásquez, and Francisco Ángel Becerra Lois. Multineutrosophic analysis of the relationship between survival and business growth in the manufacturing sector of azuay province, 2020–2023, using plithogenic n-superhypergraphs. *Neutrosophic Sets and Systems*, 84:341–355, 2025.

[38] Julio Cesar Méndez Bravo, Claudia Jeaneth Bolanos Piedrahita, Manuel Alberto Méndez Bravo, and Luis Manuel Pilacuan-Bonete. Integrating smed and industry 4.0 to optimize processes with plithogenic n-superhypergraphs. *Neutrosophic Sets and Systems*, 84:328–340, 2025.

[39] Mohammed Alqahtani. Intuitionistic fuzzy quasi-supergraph integration for social network decision making. *International Journal of Analysis and Applications*, 23:137–137, 2025.

[40] Berrocal Villegas Salomón Marcos, Montalvo Fritas Willner, Berrocal Villegas Carmen Rosa, Flores Fuentes Rivera María Yissel, Espejo Rivera Roberto, Laura Daysi Bautista Puma, and Dante Manuel Macazana Fernández. Using plithogenic n-superhypergraphs to assess the degree of relationship between information skills and digital competencies. *Neutrosophic Sets and Systems*, 84:513–524, 2025.

[41] Takaaki Fujita. Hypergraph and superhypergraph approaches in electronics: A hierarchical framework for modeling power-grid hypernetworks and superhypernetworks. *Journal of Energy Research and Reviews*, 17(6):102–136, 2025.

[42] Shouxian Zhu. Neutrosophic n-superhypernetwork: A new approach for evaluating short video communication effectiveness in media convergence. *Neutrosophic Sets and Systems*, 85:1004–1017, 2025.

[43] Alain Bretto. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*, 1, 2013.

[44] Florentin Smarandache. *SuperHyperFunction, SuperHyperStructure, Neutrosophic SuperHyperFunction and Neutrosophic SuperHyperStructure: Current understanding and future directions*. Infinite Study, 2023.

[45] Ajoy Kanti Das, Rajat Das, Suman Das, Bijoy Krishna Debnath, Carlos Granados, Bimal Shil, and Rakhal Das. A comprehensive study of neutrosophic superhyper bci-semigroups and their algebraic significance. *Transactions on Fuzzy Sets and Systems*, 8(2):80, 2025.

[46] Takaaki Fujita. Rethinking strategic perception: Foundations and advancements in hypergame theory and superhypergame theory. *Prospects for Applied Mathematics and Data Analysis*, 4(2):01–14, 2024.

[47] Florentin Smarandache. Superhyperstructure & neutrosophic superhyperstructure, 2024. Accessed: 2024-12-01.

[48] Florentin Smarandache. *Extension of HyperGraph to n-SuperHyperGraph and to Plithogenic n-SuperHyperGraph, and Extension of HyperAlgebra to n-ary (Classical-/Neutro-/Anti-) HyperAlgebra*. Infinite Study, 2020.

[49] Takaaki Fujita. Review of probabilistic hypergraph and probabilistic superhypergraph. *Spectrum of Operational Research*, 3(1):319–338, 2025.

[50] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[51] Z Nazari and B Mosapour. The entropy of hyperfuzzy sets. *Journal of Dynamical Systems and Geometric Theories*, 16(2):173–185, 2018.

[52] Young Bae Jun, Kul Hur, and Kyoung Ja Lee. Hyperfuzzy subalgebras of bck/bci-algebras. *Annals of Fuzzy Mathematics and Informatics*, 2017.

[53] Jayanta Ghosh and Tapas Kumar Samanta. Hyperfuzzy sets and hyperfuzzy group. *Int. J. Adv. Sci. Technol*, 41:27–37, 2012.

[54] Krassimir Atanassov and George Gargov. Elements of intuitionistic fuzzy logic. part i. *Fuzzy sets and systems*, 95(1):39–52, 1998.

[55] Bui Cong Cuong and Vladik Kreinovich. Picture fuzzy sets-a new concept for computational intelligence problems. In *2013 third world congress on information and communication technologies (WICT 2013)*, pages 1–6. IEEE, 2013.

[56] Vicenç Torra and Yasuo Narukawa. On hesitant fuzzy sets and decision. In *2009 IEEE international conference on fuzzy systems*, pages 1378–1382. IEEE, 2009.

[57] Florentin Smarandache. A unifying field in logics: Neutrosophic logic. In *Philosophy*, pages 1–141. American Research Press, 1999.

[58] Said Broumi, Mohamed Talea, Assia Bakali, and Florentin Smarandache. Single valued neutrosophic graphs. *Journal of New theory*, (10):86–101, 2016.

[59] Said Broumi, Mohamed Talea, Assia Bakali, and Florentin Smarandache. Interval valued neutrosophic graphs. *Critical Review, XII*, 2016:5–33, 2016.

[60] R Radha, A Stanis Arul Mary, and Florentin Smarandache. Quadripartitioned neutrosophic pythagorean soft set. *International Journal of Neutrosophic Science (IJNS) Volume 14, 2021*, page 11, 2021.

[61] Florentin Smarandache. Plithogeny, plithogenic set, logic, probability, and statistics. *arXiv preprint arXiv:1808.03948*, 2018.

[62] Muhammad Azeem, Humera Rashid, Muhammad Kamran Jamil, Selma Gütmen, and Erfan Babaee Tirkolaee. Plithogenic fuzzy graph: A study of fundamental properties and potential applications. *Journal of Dynamics and Games*, pages 0–0, 2024.