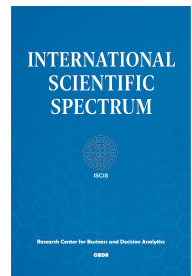




International Scientific Spectrum

Journal homepage: www.iscis-journal.org
ISSN: 3104-3305



Extensions of Factor, Signal-Flow, Control-Flow, Dataflow, and PERT/CPM Graphs to Hypergraphs and Superhypergraphs with Illustrative Applications in Engineering and Intelligent Computing

Takaaki Fujita^{1,*}, Ajoy Kanti Das², Arif Mehmood³, Arkan A Ghaib⁴

¹Independent Researcher, Tokyo, Japan

²Department of Mathematics, Tripura University, Agartala-799022, Tripura, India

³Department of Mathematics, Institute of Numerical Sciences, Gomal University, Dera Ismail Khan 29050, KPK, Pakistan

⁴Department of Information Technology, Management Technical College, Southern Technical University, Basrah, 61004, Iraq

ARTICLE INFO

Article history:

Received 14 June 2026

Received in revised form 22 August 2026

Accepted 26 September 2026

Available online 03 October 2026

Keywords:

Superhypergraph; Hypergraph; Factor graph; Signal flow graph; Control flow graph; Dataflow graph; PERT/CPM

ABSTRACT

Graphs represent pairwise relations through vertices and edges, whereas hypergraphs capture multi-way interactions by allowing each hyperedge to connect an arbitrary number of vertices. Superhypergraphs extend this representation through an iterated powerset construction, enabling nested, multi-level relations in which set-valued objects serve as higher-order entities. This paper develops a unified framework for extending five widely used graph models—factor graphs, signal flow graphs, control flow graphs, dataflow graphs, and PERT/CPM networks—to hypergraphs and superhypergraphs. For each model, we present a uniform lifting principle, formal definitions of the corresponding hypergraph and n -superhypergraph structures, and illustrative examples showing how multi-way interactions and hierarchical dependencies can be represented. We also establish that the classical graph models are embedded as special cases of the proposed frameworks. These constructions provide a theoretical foundation for multi-level dependency modeling across inference, dynamical systems, program analysis, streaming computation, and project scheduling, with computational implementation and empirical validation remaining directions for future research.

1. Introduction

1.1 From Graphs to SuperHyperGraphs

Traditional graphs model pairwise relations by representing entities as vertices and their interactions as edges. Hypergraphs extend this framework by allowing a single *hyperedge* to connect an arbitrary nonempty subset of vertices, thereby capturing genuine higher-order interactions [1–3]. Related extensions, such as Fuzzy HyperGraphs [4–6], have also been extensively studied. SuperHyperGraphs go one step further: by iterating the powerset construction, set-valued objects formed at one level may be treated as vertices at the next, producing a nested, multi-level network that naturally encodes hierarchical or embedded dependencies [7–9]. This layered formalism therefore provides a unified language for modeling and analyzing complex systems whose relationships are not purely pairwise and whose structure is inherently multi-level [10, 11].

*Corresponding author..

E-mail address: Takaaki.fujita060@gmail.com

<https://doi.org/10.66972/iscis21202627>

© The Author(s) 2026 | [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/)

Table 1 summarizes the key distinctions among graphs, hypergraphs, and superhypergraphs. Throughout this book, n denotes a natural number unless stated otherwise.

Table 1
Key distinctions among graph, hypergraph, and superhypergraph

Concept	Notation	Edge Type	Extension Mechanism
Graph [12]	$G = (V, E)$	$E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$	Standard edges connect exactly two vertices.
Hypergraph [13]	$H = (V, E)$	$E \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$	Hyperedges may join any nonempty subset of vertices.
Superhypergraph [14]	$\text{SHG}^{(n)} = (V_0, V, E)$	$V \subseteq \mathcal{P}^n(V_0), \quad E \subseteq \mathcal{P}(V)$	Applies an n -fold power-set construction to encode nested structure.

Notation. $\mathcal{P}(X) = \{A \mid A \subseteq X\}$ and $\mathcal{P}^0(X) = X, \mathcal{P}^{k+1}(X) = \mathcal{P}(\mathcal{P}^k(X))$.

1.2 Graph Paradigms Across Disciplines

Graph-based abstractions appear throughout science and engineering. In this paper we focus on five canonical paradigms that repeatedly arise in practice:

- **Factor graphs:** bipartite graphs that connect variable nodes to factor nodes, expressing a global function as a product of local factors [15–17].
- **Signal flow graphs:** directed graphs in which nodes represent signal variables and weighted edges represent gains, enabling systematic derivation of transfer functions [18, 19].
- **Control flow graphs:** directed graphs whose vertices correspond to basic blocks and whose edges represent possible transfers of control, forming a core tool in program analysis, optimization, and testing [20, 21].
- **Dataflow graphs:** directed graphs in which vertices are computational actors and edges are token channels, capturing data dependencies for parallel execution and scheduling [22–24].
- **PERT/CPM networks:** directed acyclic graphs with events as vertices and activities as edges, supporting critical-path analysis and project scheduling [25].

1.3 Our Contributions

We show how each of the above paradigms can be lifted in a systematic way to the settings of hypergraphs and superhypergraphs. For every model, we (i) define a corresponding hypergraph formalism, (ii) introduce an n -SuperHyperGraph extension that captures hierarchical structure, and (iii) provide concrete examples demonstrating the additional expressiveness afforded by hyperedges and supervertices. These constructions yield a unified foundation for structured, multi-level dependency modeling across a broad range of applications.

2. Preliminaries

Unless stated otherwise, all graphs in this paper are assumed to be finite and simple. This section fixes notation and recalls the basic definitions needed in the sequel; for further background, we refer to the cited references.

2.1 Hypergraphs and SuperHyperGraphs

Hypergraphs extend ordinary graphs by allowing a single edge to connect an arbitrary number of vertices, thereby capturing higher-order interactions [26]. SuperHyperGraphs generalize this idea by iterating the powerset construction, which enables nested, multi-level vertex and edge objects and thus supports hierarchical dependencies [27, 28]. Such frameworks have been used in diverse settings, including molecular modeling, network analysis, and signal processing [29, 30].

Definition 2.1 (Base set). A *base set* is a nonempty finite set S . All subsequent constructions are built from S .

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

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

Definition 2.3 (Hypergraph). [13, 31] A *hypergraph* is a pair $H = (V, E)$ consisting of

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

Thus, unlike an ordinary graph, a hyperedge may join any number of vertices.

Definition 2.4 (Iterated powerset). [32, 33] For $k \geq 0$, define the iterated powersets of S recursively by

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

Hence $\mathcal{P}^k(S)$ is obtained by applying the powerset operator k times.

Definition 2.5 (n -SuperHyperGraph). [14, 34] Let $n \geq 1$ and let S be a base set. An *n -superhypergraph* is a pair

$$\text{SHG}^{(n)} = (V, E), \quad V \subseteq \mathcal{P}^n(S), \quad E \subseteq \mathcal{P}(V) \setminus \{\emptyset\}.$$

Elements of V are called *n -supervertices*, and elements of E are called *n -superedges*. Intuitively, supervertices are set-valued objects at level n , and superedges relate collections of such objects, thereby enabling nested and hierarchical connectivity.

Example 2.6 (Corporate hierarchy as a 3-SuperHyperGraph). Let the base set $S = V_0$ consist of eight employees:

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

Level 1 (teams). Form three project teams:

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

Then

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

Level 2 (departments). Group teams into departments:

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

so that

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

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

$$\Delta_{\text{north}} = \{D_{\text{eng}}\}, \quad \Delta_{\text{south}} = \{D_{\text{mkt}}\}, \quad \Delta_{\text{all}} = \{D_{\text{eng}}, D_{\text{mkt}}\},$$

and set

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

Superedges. At the top level, company-wide coordination involves both divisions; we encode this by the superedge

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

Therefore,

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

is a 3-superhypergraph (Definition 2.5) encoding the four-tier hierarchy

$$\text{employees} \rightarrow \text{teams} \rightarrow \text{departments} \rightarrow \text{divisions}.$$

Figure 1 summarizes the structure of this example.

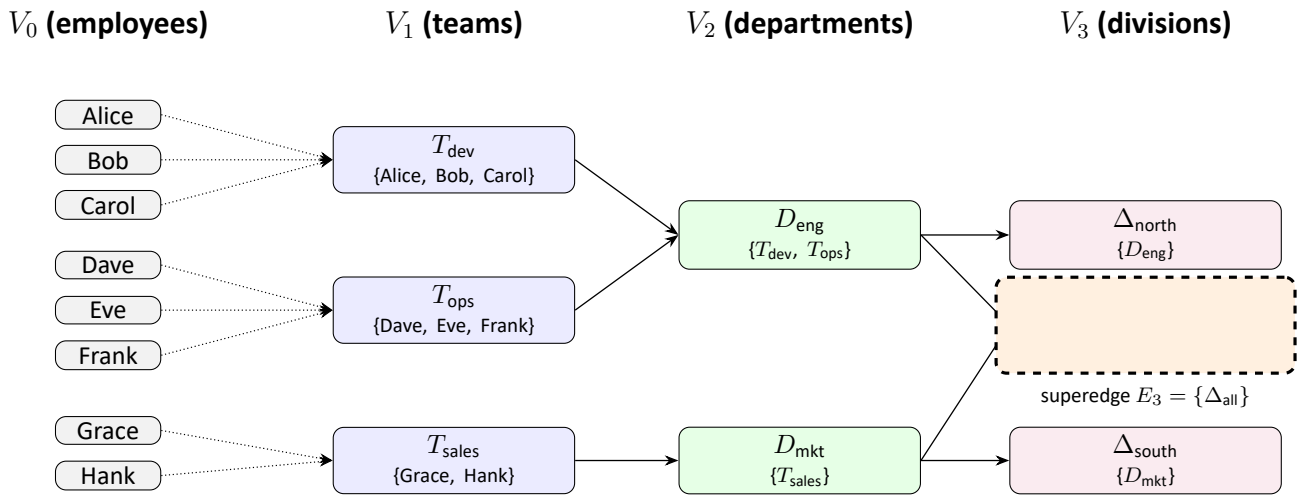


Fig. 1. A corporate hierarchy encoded as a 3-SuperHyperGraph (Example 2.6). Dotted arrows indicate membership $V_0 \rightarrow V_1$, solid arrows indicate aggregation across levels, and the top-level superedge is $E_3 = \{\Delta_{\text{all}}\}$.

3. Factor Graphs and Their Extensions

Factor graphs provide a standard bipartite representation of a product decomposition, linking *variable nodes* to *factor nodes* so that a global nonnegative function factors into a product of local terms (cf. [15–17]). In this section, we recast factor graphs within the broader settings of hypergraphs and superhypergraphs, which naturally support multi-way and hierarchical dependencies.

Definition 3.1 (Factor graph). (cf. [15, 16]) A *factor graph* consists of

- two finite disjoint sets V and F , called the sets of *variable nodes* and *factor nodes*, respectively,
- a set of edges $E \subseteq V \times F$ (so the underlying graph is bipartite), and
- for each $v \in V$, a nonempty domain $\mathcal{X}_v$, and for each $f \in F$, a *factor function*

$$\varphi_f : \mathcal{X}_{N(f)} \rightarrow \mathbb{R}_{\geq 0}, \quad N(f) := \{v \in V : (v, f) \in E\} \neq \emptyset,$$

where $\mathcal{X}_S := \prod_{v \in S} \mathcal{X}_v$ for any $S \subseteq V$.

The associated *global function* $\Phi : \mathcal{X}_V \rightarrow \mathbb{R}_{\geq 0}$ is

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

Example 3.2 (3-bit parity code as a factor graph). Let the variable set be $V = \{x_1, x_2, x_3, x_4\}$, with $\mathcal{X}_{x_i} = \{0, 1\}$ for each i . Introduce two parity-check factors $F = \{f_1, f_2\}$ with scopes

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

Define

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

The edge set is $E = \{(v, f) \in V \times F : v \in N(f)\}$. Then the global function is

$$\Phi(x_1, x_2, x_3, x_4) = \varphi_{f_1}(x_1, x_2, x_3) \varphi_{f_2}(x_2, x_3, x_4),$$

which enforces both parity constraints simultaneously.

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

$$H = (V, \mathcal{E}, \{\psi_e\}_{e \in \mathcal{E}}),$$

where V is a finite set of variables, $\mathcal{E}$ is a finite *family* of nonempty subsets of V (i.e., hyperedges; allowing repetition if different factors share the same scope), and each hyperedge $e \in \mathcal{E}$ carries a factor function

$$\psi_e : \mathcal{X}_e \rightarrow \mathbb{R}_{\geq 0}, \quad \mathcal{X}_e := \prod_{v \in e} \mathcal{X}_v.$$

The associated global function $\Psi : \mathcal{X}_V \rightarrow \mathbb{R}_{\geq 0}$ is

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

Example 3.4 (3-bit parity code as a factor hypergraph). With $V = \{x_1, x_2, x_3, x_4\}$ as in Example 3.2, take the hyperedge family

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

Set $\psi_{e_i} := \varphi_{f_i}$ for $i = 1, 2$, viewing φ_{f_i} as a function on $\mathcal{X}_{e_i}$. Then

$$\Psi(x_1, x_2, x_3, x_4) = \psi_{e_1}(x_1, x_2, x_3) \psi_{e_2}(x_2, x_3, x_4)$$

coincides with Φ .

Theorem 3.5 (Factor hypergraphs underlie hypergraphs). Let $H = (V, \mathcal{E}, \{\psi_e\}_{e \in \mathcal{E}})$ be a factor hypergraph. Then $(V, \mathcal{E})$ is an (undirected) hypergraph in the sense of Definition 2.3.

Proof. By Definition 3.3, each $e \in \mathcal{E}$ is a nonempty subset of V , i.e., $\mathcal{E} \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$. Hence $(V, \mathcal{E})$ is a hypergraph. $\square$

Theorem 3.6 (Factor graphs embed into factor hypergraphs). *Let $G = (V, F, E, \{\varphi_f\}_{f \in F})$ be a factor graph as in Definition 3.1. Define a factor hypergraph*

$$H = (V, \mathcal{E}, \{\psi_e\}_{e \in \mathcal{E}})$$

by indexing hyperedges by factors:

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

Then the global functions coincide: $\Psi = \Phi$ on $\mathcal{X}_V$.

Proof. For each $f \in F$, the neighborhood $N(f)$ is nonempty, so $e_f = N(f)$ is a valid hyperedge. For any $x \in \mathcal{X}_V$,

$$\Psi(x) = \prod_{e \in \mathcal{E}} \psi_e(x_e) = \prod_{f \in F} \psi_{e_f}(x_{e_f}) = \prod_{f \in F} \varphi_f(x_{N(f)}) = \Phi(x).$$

□

Definition 3.7 (Factor n -superhypergraph). Let V_0 be a finite set of *atomic variables*, and let each $v \in V_0$ take values in a nonempty domain $\mathcal{X}_v$. For $k \geq 0$, define iterated powersets by

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

Define a *flattening map* $\flat : \mathcal{P}^n(V_0) \rightarrow \mathcal{P}(V_0)$ recursively by

$$\flat(v) = \{v\} \quad (v \in V_0), \quad \flat(A) = \bigcup_{u \in A} \flat(u) \quad (A \in \mathcal{P}^{k+1}(V_0)).$$

A *factor n -superhypergraph* is a triple

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

where $V_n \subseteq \mathcal{P}^n(V_0)$ is a set of n -supervertices, $E_n \subseteq \mathcal{P}(V_n) \setminus \{\emptyset\}$ is a set of n -superedges, and each $e \in E_n$ is equipped with a factor function

$$\psi_e : \mathcal{X}_{\flat(e)} \rightarrow \mathbb{R}_{\geq 0}, \quad \flat(e) := \bigcup_{U \in e} U \subseteq V_0, \quad \mathcal{X}_{\flat(e)} := \prod_{v \in \flat(e)} \mathcal{X}_v.$$

The induced global function $\Psi : \mathcal{X}_{V_0} \rightarrow \mathbb{R}_{\geq 0}$ is

$$\Psi(x) = \prod_{e \in E_n} \psi_e(x_{\flat(e)}), \quad x \in \mathcal{X}_{V_0}.$$

Example 3.8 (3-bit parity code as a factor 1-superhypergraph). Let $V_0 = \{x_1, x_2, x_3, x_4\}$ with $\mathcal{X}_{x_i} = \{0, 1\}$. Let

$$U_1 = \{x_1, x_2, x_3\}, \quad U_2 = \{x_2, x_3, x_4\}, \quad V_1 = \{U_1, U_2\} \subseteq \mathcal{P}(V_0).$$

Define a single superedge $e = \{U_1, U_2\} \in \mathcal{P}(V_1)$, so $E_1 = \{e\}$. Let $\psi_e : \mathcal{X}_{\flat(e)} \rightarrow \mathbb{R}_{\geq 0}$ with $\flat(e) = U_1 \cup U_2$ be

$$\psi_e(x_1, x_2, x_3, x_4) = \psi_{U_1}(x_1, x_2, x_3) \psi_{U_2}(x_2, x_3, x_4),$$

where ψ_{U_1} and ψ_{U_2} are the parity-check functions from Example 3.4. Then $\mathcal{F}^{(1)} = (V_1, E_1, \{\psi_e\})$ encodes both parity checks via a single superedge.

Theorem 3.9 (Underlying n -superhypergraph). *If $\mathcal{F}^{(n)} = (V_n, E_n, \{\psi_e\}_{e \in E_n})$ is a factor n -superhypergraph, then the pair (V_n, E_n) is an n -superhypergraph in the structural sense that*

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

Proof. This is immediate from Definition 3.7 upon forgetting the factor functions $\{\psi_e\}$. □

Theorem 3.10 (Factor hypergraphs embed as factor 1-superhypergraphs). *Let $H = (V, E, \{\psi_e\}_{e \in E})$ be a factor hypergraph (Definition 3.3). Define $V_0 := V$ and the singleton injection $\iota : V \rightarrow \mathcal{P}(V)$ by $\iota(v) = \{v\}$. Set*

$$V_1 := \iota(V) = \{\{v\} : v \in V\} \subseteq \mathcal{P}(V_0), \quad E_1 := \{\iota(e) : e \in E\} \subseteq \mathcal{P}(V_1) \setminus \{\emptyset\},$$

where $\iota(e) := \{\{v\} : v \in e\}$. For each $e \in E$, define $\psi'_{\iota(e)} : \mathcal{X}_e \rightarrow \mathbb{R}_{\geq 0}$ by

$$\psi'_{\iota(e)}(x_e) := \psi_e(x_e), \quad x \in \mathcal{X}_V.$$

Then $\mathcal{F}^{(1)} := (V_1, E_1, \{\psi'_{e'}\}_{e' \in E_1})$ is a factor 1-superhypergraph whose global function equals that of H .

Proof. Each $\iota(e)$ is nonempty because $e \neq \emptyset$, hence $E_1 \subseteq \mathcal{P}(V_1) \setminus \{\emptyset\}$. Moreover, for $e' = \iota(e)$ we have $b(e') = e$, so $\psi'_{e'}$ is well-defined on $\mathcal{X}_{b(e')} = \mathcal{X}_e$. Finally, for any $x \in \mathcal{X}_V$,

$$\prod_{e' \in E_1} \psi'_{e'}(x_{b(e')}) = \prod_{e \in E} \psi'_{\iota(e)}(x_e) = \prod_{e \in E} \psi_e(x_e),$$

so the factorization (and thus the global function) is preserved. □

4. Signal Flow Graphs and Their Extensions

Signal flow graphs are widely used to represent linear signal dependencies in engineering systems: nodes carry signal variables, and directed branches carry gains that quantify how an upstream signal contributes to a downstream signal (cf. [18, 19]). In this section, we extend this paradigm to *signal flow hypergraphs* and further to *signal flow n -superhypergraphs*, which allow multi-input dependencies and hierarchical (multi-level) aggregation in a mathematically uniform way.

Definition 4.1 (Signal flow graph). (cf. [18, 19]) 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*,
- $\gamma : E \rightarrow \mathbb{R}$ assigns a real-valued *gain* to each branch.

Attach to each node $v \in V$ a signal variable $x_v \in \mathbb{R}$ and an external input $\eta_v \in \mathbb{R}$. The node equations are

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

i.e., the signal at v is the sum of gain-weighted incoming contributions plus an external input.

Example 4.2 (Feedback amplifier as a signal flow graph). Consider three signal nodes

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

with directed branches

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

and gains

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

Then the node equations become

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

which recovers the standard linear feedback relation (up to external inputs).

Definition 4.3 (Signal flow hypergraph). A *signal flow hypergraph* is a quintuple $H = (V, \mathcal{E}, t, h, \gamma)$, where

- V is a finite set of nodes,
- $\mathcal{E}$ is a finite set of *hyperbranches*,
- $t : \mathcal{E} \rightarrow \mathcal{P}(V) \setminus \{\emptyset\}$ assigns to each hyperbranch e a nonempty *tail set* $t(e)$,
- $h : \mathcal{E} \rightarrow V$ assigns to each hyperbranch e a single *head node* $h(e)$,
- $\gamma : \mathcal{E} \rightarrow \mathbb{R}$ assigns a *gain* to each hyperbranch.

Attach to each $v \in V$ a signal variable $x_v \in \mathbb{R}$ and an external input $\eta_v \in \mathbb{R}$. The node equations are

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

Thus a single hyperbranch can aggregate signals from multiple tail nodes into one head node.

Example 4.4 (Sensor fusion as a signal flow hypergraph). Let

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

be three sensor nodes and one fusion node. Define a single hyperbranch $e \in \mathcal{E}$ by

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

Then the fusion equation becomes

$$x_F = \frac{1}{3}(x_{S_1} + x_{S_2} + x_{S_3}) + \eta_F,$$

which encodes equal-weight averaging of three inputs via one multi-input hyperbranch.

Theorem 4.5 (Underlying hypergraph). Let $H = (V, \mathcal{E}, t, h, \gamma)$ be a signal flow hypergraph. Then

$$H_{\text{und}} = (V, E_{\text{und}}), \quad E_{\text{und}} := \{t(e) \cup \{h(e)\} \mid e \in \mathcal{E}\},$$

is an (undirected) hypergraph.

Proof. For each $e \in \mathcal{E}$, the tail $t(e)$ is nonempty and $h(e) \in V$, hence $t(e) \cup \{h(e)\} \subseteq V$ is a nonempty subset of V . Therefore $E_{\text{und}} \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$, so (V, E_{und}) is a hypergraph. $\square$

Theorem 4.6 (Signal flow graphs embed into signal flow hypergraphs). *Every signal flow graph $G = (V, E_G, \gamma_G)$ (Definition 4.1) induces a signal flow hypergraph $H = (V, \mathcal{E}, t, h, \gamma)$ by setting*

$$\mathcal{E} := \{e_{u,v} \mid (u, v) \in E_G\}, \quad t(e_{u,v}) = \{u\}, \quad h(e_{u,v}) = v, \quad \gamma(e_{u,v}) = \gamma_G(u, v).$$

Under the identification $e_{u,v} \leftrightarrow (u, v)$, the node equations of H coincide with those of G .

Proof. For any $v \in V$,

$$\sum_{\substack{e \in \mathcal{E} \\ h(e)=v}} \gamma(e) \sum_{u \in t(e)} x_u = \sum_{(u,v) \in E_G} \gamma_G(u, v) x_u,$$

because each hyperbranch with head v has tail $\{u\}$. Hence the node equations match term-by-term. $\square$

Definition 4.7 (Signal flow n -superhypergraph). Let V_0 be a finite set of *atomic signal nodes*. For $k \geq 0$, define iterated powersets by

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

A *signal flow n -superhypergraph* is a quintuple

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

where

- $V_n \subseteq \mathcal{P}^n(V_0)$ is a set of n -supervertices,
- $E_n \subseteq \mathcal{P}(V_n) \setminus \{\emptyset\}$ is a set of n -superedges,
- $t : E_n \rightarrow \mathcal{P}(V_n) \setminus \{\emptyset\}$ assigns each superedge a nonempty *tail set*,
- $h : E_n \rightarrow V_n$ assigns each superedge a single *head supervertex*,
- $\gamma : E_n \rightarrow \mathbb{R}$ assigns a gain to each superedge.

Attach to each $v \in V_n$ a signal variable $x_v \in \mathbb{R}$ and an external input $\eta_v \in \mathbb{R}$. The supervertex equations are

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

Example 4.8 (Hierarchical signal processing as a signal flow 2-superhypergraph). Let the atomic nodes be

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

Define two 1-supervertices (modules)

$$v_A = \{S_1, S_2, S_3\}, \quad v_B = \{C\}, \quad V_1 = \{v_A, v_B\} \subseteq \mathcal{P}(V_0).$$

At level 2, consider the 2-supervertices

$$U_A = \{v_A\}, \quad U_B = \{v_B\}, \quad v_{\text{sys}} = \{v_A, v_B\}, \quad V_2 = \{U_A, U_B, v_{\text{sys}}\} \subseteq \mathcal{P}^2(V_0).$$

Define two superedges $e_1, e_2 \in \mathcal{P}(V_2) \setminus \{\emptyset\}$ by

$$t(e_1) = \{U_A\}, \quad h(e_1) = U_B, \quad \gamma(e_1) = 1, \quad t(e_2) = \{U_A, U_B\}, \quad h(e_2) = v_{\text{sys}}, \quad \gamma(e_2) = 1,$$

and set $E_2 = \{e_1, e_2\}$. Then $\text{SFSH}^{(2)} = (V_2, E_2, t, h, \gamma)$ encodes a two-level pipeline in which (i) the control module depends on the fused sensing module, and (ii) the system-level output depends jointly on both modules.

Figure 2 summarizes the structure of this example.

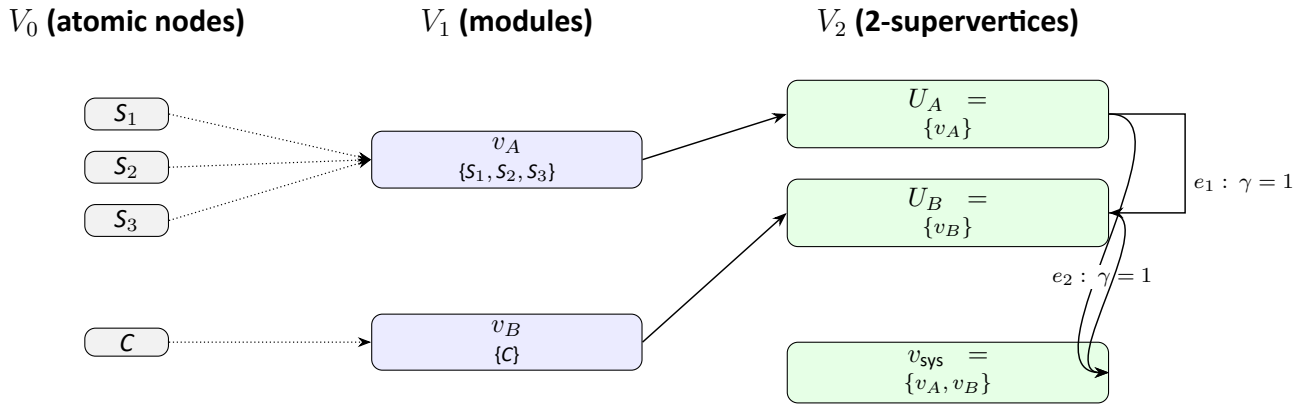


Fig. 2. Hierarchical signal processing encoded as a signal flow 2-superhypergraph (Example 4.8).

Theorem 4.9 (Underlying n -superhypergraph). *If*

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

is a signal flow n -superhypergraph, then the pair (V_n, E_n) is an n -superhypergraph in the structural sense of Definition 2.5.

Proof. By Definition 4.7, $V_n \subseteq \mathcal{P}^n(V_0)$ and $E_n \subseteq \mathcal{P}(V_n) \setminus \{\emptyset\}$. Forgetting t, h, γ therefore yields the required n -superhypergraph structure on (V_n, E_n) . $\square$

Theorem 4.10 (Signal flow hypergraphs embed as signal flow 1-superhypergraphs). *Let $H = (V, \mathcal{E}, t, h, \gamma)$ be a signal flow hypergraph (Definition 4.3). Set $V_0 := V$ and define the singleton injection $\iota : V \rightarrow \mathcal{P}(V)$ by $\iota(v) = \{v\}$. Let*

$$V_1 := \iota(V) = \{\{v\} \mid v \in V\} \subseteq \mathcal{P}(V_0), \quad E_1 := \{\iota_e(e) \mid e \in \mathcal{E}\} \subseteq \mathcal{P}(V_1) \setminus \{\emptyset\},$$

where $\iota_e(e)$ is the superedge corresponding to e . Define

$$\iota_e(e) := \{\iota(u) \mid u \in t(e)\} \subseteq V_1, \quad t'(\iota_e(e)) := \{\iota(u) \mid u \in t(e)\}, \quad h'(\iota_e(e)) := \iota(h(e)),$$

and $\gamma'(\iota_e(e)) := \gamma(e)$. Then $(V_1, E_1, t', h', \gamma')$ is a signal flow 1-superhypergraph whose node equations coincide with those of H under the identification $x_{\{v\}} = x_v$.

Proof. Each $t(e)$ is nonempty, hence $\iota_e(e) \neq \emptyset$ and so $E_1 \subseteq \mathcal{P}(V_1) \setminus \{\emptyset\}$. Moreover, t' and h' satisfy the nonempty-tail and single-head conditions by construction, and γ' is well-defined. Finally, using $x_{\{v\}} = x_v$,

$$\sum_{\substack{e' \in E_1 \\ h'(e') = \{v\}}} \gamma'(e') \sum_{u' \in t'(e')} x_{u'} = \sum_{\substack{e \in \mathcal{E} \\ h(e) = v}} \gamma(e) \sum_{u \in t(e)} x_u,$$

so the signal equations are preserved. $\square$

5. Control Flow Graphs and Their Extensions

Control flow graphs (CFGs) are a standard abstraction of program execution: vertices represent program points (e.g., basic blocks) and directed edges represent possible transfers of control (cf. [20, 21, 35]). To capture multi-way branching/joining as single primitives and to support hierarchical program structure (e.g., functions, modules, call graphs), we extend CFGs to *control flow hypergraphs* and further to *control flow n -superhypergraphs*.

Definition 5.1 (Control flow graph). (cf. [35]) A *control flow graph* is a quadruple

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

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*,
- $\text{entry} \in N$ is the distinguished *entry* node, and $\text{exit} \in N$ is the distinguished *exit* node,

satisfying the following conditions:

- (i) *entry* has no predecessors: there is no $u \in N$ with $(u, \text{entry}) \in E$.
- (ii) *exit* has no successors: there is no $v \in N$ with $(\text{exit}, v) \in E$.
- (iii) (Reachability) Every node is reachable from *entry*, and *exit* is reachable from every node.

Example 5.2 (If–else statement as a control flow graph). Consider the pseudo-code:

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

A corresponding CFG $G = (N, E, \text{entry}, \text{exit})$ is obtained by taking

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

and

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

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

Definition 5.3 (Control flow hypergraph). A *control flow hypergraph* is a quadruple

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

where

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

In addition, H is equipped with distinguished nodes $\text{entry}, \text{exit} \in N$ satisfying the reachability conditions analogous to Definition 5.1: every node is reachable from *entry* via directed hyperpaths and can reach *exit* via directed hyperpaths.

Example 5.4 (Switch–case as a control flow hypergraph). Consider the schematic switch–case construct:

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

Let

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

Define two hyperedges $e_1, e_2 \in \mathcal{E}$ by

$$t(e_1) = \{\text{switch}\}, \quad h(e_1) = \{v_A, v_B, v_{\text{def}}\},$$

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

Then $H = (N, \{e_1, e_2\}, t, h)$ models a multi-way branch (from *switch*) followed by a join at v_{merge} .

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

$$H_{\text{und}} = (N, E_{\text{und}}), \quad E_{\text{und}} := \{t(e) \cup h(e) \mid e \in \mathcal{E}\},$$

is an (undirected) hypergraph.

Proof. For each $e \in \mathcal{E}$, both $t(e)$ and $h(e)$ are nonempty subsets of N , hence $t(e) \cup h(e) \subseteq N$ is nonempty. Therefore $E_{\text{und}} \subseteq \mathcal{P}(N) \setminus \{\emptyset\}$, so (N, E_{und}) is a hypergraph. $\square$

Theorem 5.6 (CFGs embed into control flow hypergraphs). *Every control flow graph $G = (N, E, \text{entry}, \text{exit})$ induces a control flow hypergraph $H = (N, \mathcal{E}, t, h)$ by setting*

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

With entry and exit unchanged, the reachability conditions are preserved.

Proof. Each edge $(u, v) \in E$ yields a hyperedge $e_{u,v}$ with nonempty tail and head. Directed paths in G correspond to directed hyperpaths in H under the bijection $(u, v) \leftrightarrow e_{u,v}$. Hence reachability from *entry* and reachability of *exit* are preserved. $\square$

Definition 5.7 (Control flow n -superhypergraph). Let V_0 be a finite set of *atomic program points* (e.g., basic blocks). For $k \geq 0$, define

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

A control flow n -superhypergraph is a quadruple

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

where

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

- $t : E_n \rightarrow \mathcal{P}(V_n) \setminus \{\emptyset\}$ assigns each $e \in E_n$ a nonempty tail set $t(e) \subseteq V_n$,
- $h : E_n \rightarrow \mathcal{P}(V_n) \setminus \{\emptyset\}$ assigns each $e \in E_n$ a nonempty head set $h(e) \subseteq V_n$.

As in the hypergraph case, one may specify distinguished supervertices $entry_n, exit_n \in V_n$ and require the corresponding reachability conditions at level n .

Example 5.8 (Nested function calls as a control flow 2-superhypergraph). Let

$$V_0 = \{e_foo, b_foo, x_foo, e_bar, b_bar, x_bar\}$$

be atomic program points for two functions `foo` and `bar`. Define 1-supervertices (function-level aggregates)

$$F_{foo} = \{e_foo, b_foo, x_foo\}, \quad F_{bar} = \{e_bar, b_bar, x_bar\},$$

so that $V_1 = \{F_{foo}, F_{bar}\} \subseteq \mathcal{P}(V_0)$. At level 2, set

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

and let $V_2 = \{U_{foo}, U_{bar}, U_{prog}\} \subseteq \mathcal{P}^2(V_0)$. Define two superedges $e_1, e_2 \in \mathcal{P}(V_2) \setminus \{\emptyset\}$ by

$$t(e_1) = \{U_{foo}\}, \quad h(e_1) = \{U_{bar}\},$$

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

Then e_1 encodes the call `foo` $\rightarrow$ `bar`, and e_2 encodes program completion after both functions have executed.

Figure 3 summarizes the structure of this example.

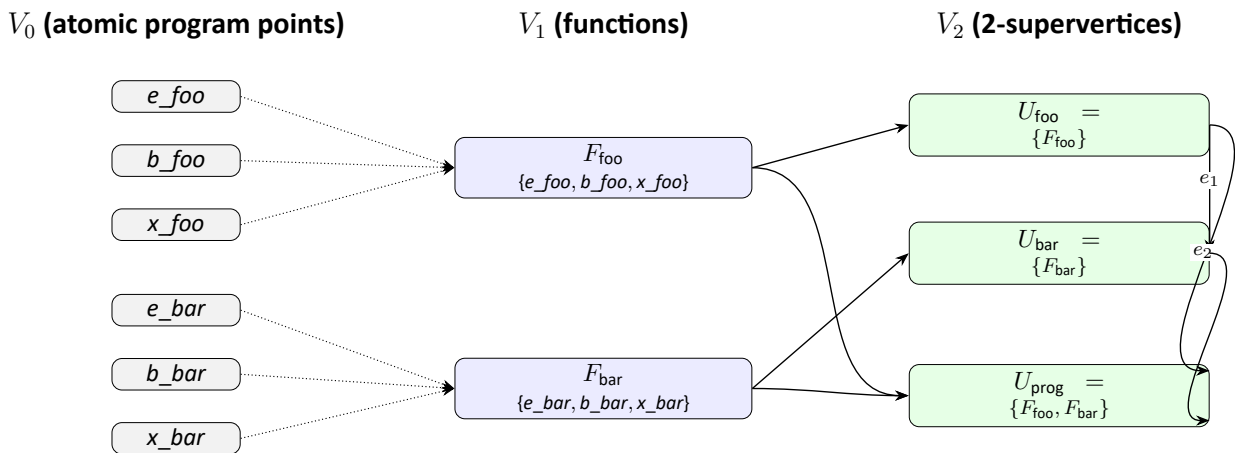


Fig. 3. Nested function calls encoded as a control flow 2-superhypergraph (Example 5.8). Dotted arrows indicate membership $V_0 \rightarrow V_1$; solid arrows indicate lifting and the level-2 superedges e_1, e_2 .

Theorem 5.9 (Underlying n -superhypergraph). *If $\text{CFGSH}^{(n)} = (V_n, E_n, t, h)$ is a control flow n -superhypergraph, then (V_n, E_n) is an n -superhypergraph in the sense of Definition 2.5.*

Proof. By Definition 5.7, $V_n \subseteq \mathcal{P}^n(V_0)$ and $E_n \subseteq \mathcal{P}(V_n) \setminus \{\emptyset\}$. Forgetting the maps t and h therefore yields an n -superhypergraph structure on (V_n, E_n) . $\square$

Theorem 5.10 (Control flow hypergraphs embed as control flow 1-superhypergraphs). *Let*

$$H = (V, \mathcal{E}, t_H, h_H)$$

be a control flow hypergraph (Definition 5.3). Set $V_0 := V$ and let $\iota : V \rightarrow \mathcal{P}(V)$ be the singleton injection $\iota(v) = \{v\}$. Define

$$V_1 := \{\iota(v) \mid v \in V\} = \{\{v\} \mid v \in V\} \subseteq \mathcal{P}(V_0), \quad E_1 := \{\iota(e) \mid e \in \mathcal{E}\} \subseteq \mathcal{P}(V_1) \setminus \{\emptyset\},$$

where $\iota(e) := \{\{u\} \mid u \in t_H(e)\} \cup \{\{v\} \mid v \in h_H(e)\}$ is the corresponding level-1 superedge. Define $t, h : E_1 \rightarrow \mathcal{P}(V_1) \setminus \{\emptyset\}$ by

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

Then $\text{CFGSH}^{(1)} = (V_1, E_1, t, h)$ is a control flow 1-superhypergraph that represents H at the singleton level.

Proof. Because $t_H(e)$ and $h_H(e)$ are nonempty for every $e \in \mathcal{E}$, both $t(\iota(e))$ and $h(\iota(e))$ are nonempty subsets of V_1 . Moreover $V_1 \subseteq \mathcal{P}(V_0)$ and $E_1 \subseteq \mathcal{P}(V_1) \setminus \{\emptyset\}$, so $\text{CFGSH}^{(1)}$ satisfies Definition 5.7 with $n = 1$. Reachability properties are preserved because hyperpaths in H correspond to hyperpaths in $\text{CFGSH}^{(1)}$ under the singleton injection. $\square$

6. Dataflow Graphs and Their Extensions

Dataflow graphs are a standard model of stream-based and parallel computation: vertices represent *actors* (computational units) and directed edges represent *channels* that carry tokens, thereby encoding data dependencies and enabling scheduling and throughput analysis (cf. [22, 23]). To represent multi-producer/multi-consumer dependencies and hierarchical (multi-level) aggregation in a single formalism, we extend dataflow graphs to *dataflow hypergraphs* and further to *dataflow n -superhypergraphs*.

Definition 6.1 (Dataflow graph). (cf. [22, 23]) A *dataflow graph* is a tuple

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

where

- V is a finite set of *actors*,
- E is a finite set of directed *channels*,
- $\text{src}, \text{dst} : E \rightarrow V$ assign to each channel $e \in E$ its unique source actor $\text{src}(e)$ and sink actor $\text{dst}(e)$,
- $\text{prd}, \text{cns} : E \rightarrow \mathbb{N}_{>0}$ give the *production* and *consumption* rates on each channel,
- $\gamma_0 : E \rightarrow \mathbb{N}$ specifies the *initial token count* on each channel.

A firing of an actor $u \in V$ consumes $\text{cns}(e)$ tokens from each incoming channel e with $\text{dst}(e) = u$ (when available) and produces $\text{prd}(e)$ tokens onto each outgoing channel e with $\text{src}(e) = u$.

Example 6.2 (Streaming FIR filter as a dataflow graph). Consider a simple streaming DSP pipeline with actors

$$V = \{\text{Read}, \text{MAC}, \text{Write}\}, \quad E = \{e_1, e_2\}.$$

Let

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

Here *Read* produces one sample token per firing, *MAC* performs multiply–accumulate on streaming inputs, and *Write* outputs the resulting token stream.

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

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

where

- V is a finite set of actors,
- E_H is a finite set of directed *hyperchannels*,
- $t, h : E_H \rightarrow \mathcal{P}(V) \setminus \{\emptyset\}$ assign to each hyperchannel $e \in E_H$ a nonempty set $t(e)$ of *producer* actors and a nonempty set $h(e)$ of *consumer* actors,
- $\text{prd}_H : E_H \times V \rightarrow \mathbb{N}_{\geq 0}$ assigns a production rate to each pair (e, u) , with $\text{prd}_H(e, u) > 0$ only if $u \in t(e)$,
- $\text{cns}_H : E_H \times V \rightarrow \mathbb{N}_{\geq 0}$ assigns a consumption rate to each pair (e, v) , with $\text{cns}_H(e, v) > 0$ only if $v \in h(e)$,
- $\gamma_0^H : E_H \rightarrow \mathbb{N}$ specifies the initial token count on each hyperchannel.

Operationally, a firing associated with a hyperchannel e consumes $\text{cns}_H(e, v)$ tokens at each $v \in h(e)$ (when available) and produces $\text{prd}_H(e, u)$ tokens at each $u \in t(e)$.

Example 6.4 (Sensor fusion as a dataflow hypergraph). Let

$$V = \{S_1, S_2, S_3, \text{Fuse}\}, \quad E_H = \{e\}.$$

Define

$$t(e) = \{S_1, S_2, S_3\}, \quad h(e) = \{\text{Fuse}\}, \quad \gamma_0^H(e) = 0,$$

and set rates

$$\text{prd}_H(e, S_i) = 1 \ (i = 1, 2, 3), \quad \text{cns}_H(e, \text{Fuse}) = 3.$$

Thus the hyperchannel e aggregates one token from each sensor and yields one fused token at *Fuse*, representing a single multi-input fusion step.

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

$$H_{\text{und}} = (V, E_{\text{und}}), \quad E_{\text{und}} := \{t(e) \cup h(e) \mid e \in E_H\},$$

is an (undirected) hypergraph.

Proof. For each $e \in E_H$, both $t(e)$ and $h(e)$ are nonempty subsets of V , so $t(e) \cup h(e) \subseteq V$ is nonempty. Hence $E_{\text{und}} \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$, and (V, E_{und}) is a hypergraph. $\square$

Theorem 6.6 (Dataflow graphs embed into dataflow hypergraphs). *Every dataflow graph*

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

induces a dataflow hypergraph

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

as follows. For each channel $e \in E$, introduce a hyperchannel $e' \in E_H$ and set

$$t(e') = \{\text{src}(e)\}, \quad h(e') = \{\text{dst}(e)\}, \quad \gamma_0^H(e') = \gamma_0(e).$$

Define rates by

$$\text{prd}_H(e', u) = \begin{cases} \text{prd}(e), & u = \text{src}(e), \\ 0, & \text{otherwise}, \end{cases} \quad \text{cns}_H(e', v) = \begin{cases} \text{cns}(e), & v = \text{dst}(e), \\ 0, & \text{otherwise}. \end{cases}$$

Then the operational semantics and token accounting coincide, so H reduces to D when each hyperchannel has singleton tail and head.

Proof. By construction, each $e' \in E_H$ has nonempty tail and head, and the production/consumption rates agree with those of e . Thus token changes induced by firings are identical in D and H under the correspondence $e \leftrightarrow e'$. $\square$

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

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

A *dataflow n -superhypergraph* is a tuple

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

where

- $V_n \subseteq \mathcal{P}^n(V_0)$ is a set of n -supervertices,
- $E_n \subseteq \mathcal{P}(V_n) \setminus \{\emptyset\}$ is a set of n -superedges,
- $t, h : E_n \rightarrow \mathcal{P}(V_n) \setminus \{\emptyset\}$ assign to each $e \in E_n$ a nonempty tail set $t(e)$ and a nonempty head set $h(e)$,
- $\text{prd}^{(n)} : E_n \times V_n \rightarrow \mathbb{N}_{\geq 0}$ and $\text{cns}^{(n)} : E_n \times V_n \rightarrow \mathbb{N}_{\geq 0}$ satisfy

$$\text{prd}^{(n)}(e, u) > 0 \Rightarrow u \in t(e), \quad \text{cns}^{(n)}(e, v) > 0 \Rightarrow v \in h(e),$$

- $\gamma_0^{(n)} : E_n \rightarrow \mathbb{N}$ specifies initial token counts.

Example 6.8 (Hierarchical image processing as a dataflow 2-superhypergraph). Let

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

Define 1-supervertices (modules)

$$M_1 = \{\text{Load}, \text{Filter}\}, \quad M_2 = \{\text{Encode}, \text{Store}\}, \quad V_1 = \{M_1, M_2\} \subseteq \mathcal{P}(V_0).$$

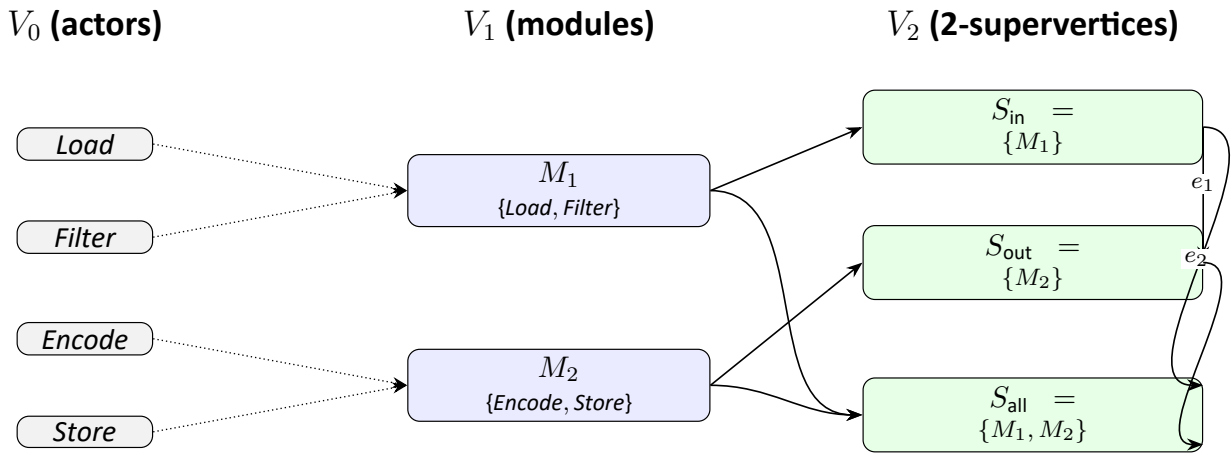


Fig. 4. Hierarchical image processing encoded as a dataflow 2-superhypergraph (Example 6.8). Dotted arrows indicate membership $V_0 \rightarrow V_1$; solid arrows indicate lifting and the level-2 superedges e_1, e_2 .

At level 2, set

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

so $V_2 = \{S_{in}, S_{out}, S_{all}\} \subseteq \mathcal{P}^2(V_0)$. Define superedges $e_1, e_2 \in \mathcal{P}(V_2) \setminus \{\emptyset\}$ by

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

Then e_1 encodes a stage-to-stage dependency, while e_2 encodes system-level completion depending jointly on both modules.

Figure 4 summarizes the structure of this example.

Theorem 6.9 (Underlying n -superhypergraph). *If $\mathcal{D}^{(n)} = (V_n, E_n, t, h, \text{prd}^{(n)}, \text{cns}^{(n)}, \gamma_0^{(n)})$ is a dataflow n -superhypergraph, then (V_n, E_n) is an n -superhypergraph in the sense of Definition 2.5.*

Proof. By Definition 6.7, $V_n \subseteq \mathcal{P}^n(V_0)$ and $E_n \subseteq \mathcal{P}(V_n) \setminus \{\emptyset\}$. Forgetting t, h and the token-rate data therefore yields the required n -superhypergraph structure on (V_n, E_n) . $\square$

Theorem 6.10 (Dataflow hypergraphs embed as dataflow 1-superhypergraphs). *Let*

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

be a dataflow hypergraph (Definition 6.3). Set $V_0 := V$ and let $\iota : V \rightarrow \mathcal{P}(V)$ be the singleton injection $\iota(v) = \{v\}$. Define

$$V_1 := \{\{v\} \mid v \in V\} \subseteq \mathcal{P}(V_0), \quad E_1 := \{\iota(e) \mid e \in E_H\} \subseteq \mathcal{P}(V_1) \setminus \{\emptyset\},$$

where $\iota(e)$ is the superedge determined by e via

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

Transport the rate and initial-token data along ι in the evident way:

$$\text{prd}^{(1)}(\iota(e), \{u\}) := \text{prd}_H(e, u), \quad \text{cns}^{(1)}(\iota(e), \{v\}) := \text{cns}_H(e, v), \quad \gamma_0^{(1)}(\iota(e)) := \gamma_0^H(e).$$

Then $\mathcal{D}^{(1)} = (V_1, E_1, t, h, \text{prd}^{(1)}, \text{cns}^{(1)}, \gamma_0^{(1)})$ is a dataflow 1-superhypergraph that faithfully represents H .

Proof. Because $t_H(e)$ and $h_H(e)$ are nonempty, the transported tail and head sets are nonempty subsets of V_1 , and the rate functions are well-defined by construction. Token updates in H and $\mathcal{D}^{(1)}$ coincide under the identification $\{v\} \leftrightarrow v$, so the embedding preserves the dataflow semantics. $\square$

7. PERT/CPM Graphs and Their Extensions

PERT/CPM networks are classical tools in project scheduling. In their standard form, they are modeled by a directed acyclic graph (DAG) whose vertices are *events* (milestones) and whose directed edges are *activities* equipped with positive durations; forward and backward passes then yield earliest/latest event times, slack values, and the critical path (cf. [25]). To express multi-predecessor constraints as single primitives and to represent hierarchical project structures, we extend PERT/CPM graphs to *PERT/CPM hypergraphs* and further to *PERT/CPM n -superhypergraphs*.

Definition 7.1 (PERT/CPM graph). (cf. [25]) A *PERT/CPM graph* is a tuple

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

where

- V is a finite set of *events*, with distinguished *start* event $s \in V$ and *finish* event $t \in V$, $s \neq t$;
- $E \subseteq V \times V$ is a set of directed *activities* such that
 - (i) $(u, v) \in E \Rightarrow u \neq v$ (no loops),
 - (ii) the digraph (V, E) is acyclic,
 - (iii) s has no predecessors and t has no successors: $(u, s) \notin E$ and $(t, v) \notin E$ for all $u, v \in V$;
- $d : E \rightarrow \mathbb{R}_{>0}$ assigns a positive *duration* to each activity.

Example 7.2 (House construction as a PERT/CPM graph). Let

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

where s denotes project start, v_1 foundation complete, v_2 framing complete, v_3 roofing complete, v_4 finishing complete, and t move-in ready. Define

$$E = \{(s, v_1), (v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, t)\},$$

with durations

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

This DAG encodes the schedule constraints for the project.

Definition 7.3 (Forward and backward pass). Let $G = (V, E, d, s, t)$ be a PERT/CPM graph. The *earliest occurrence time* (EOT) is the map $\text{EOT} : V \rightarrow \mathbb{R}_{\geq 0}$ defined by

$$\text{EOT}(s) = 0, \quad \text{EOT}(v) = \max_{(u,v) \in E} (\text{EOT}(u) + d(u, v)) \quad (v \neq s),$$

where the maximum is taken over all immediate predecessors u of v . The *latest occurrence time* (LOT) is the map $\text{LOT} : V \rightarrow \mathbb{R}_{\geq 0}$ defined by

$$\text{LOT}(t) = \text{EOT}(t), \quad \text{LOT}(v) = \min_{(v,w) \in E} (\text{LOT}(w) - d(v, w)) \quad (v \neq t),$$

where the minimum is taken over all immediate successors w of v .

Definition 7.4 (Slack times). For each activity $(u, v) \in E$, define the *total slack*

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

and the *free slack*

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

Definition 7.5 (Critical activities and critical path). An activity $(u, v) \in E$ is *critical* if $S_T(u, v) = 0$. A *critical path* is a directed s -to- t path consisting entirely of critical activities. The project duration equals $EOT(t)$.

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

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

where

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

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

Example 7.7 (New product launch as a PERT/CPM hypergraph). Let

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

Model the constraint that launch requires completion of design, prototype, testing, and marketing by a single hyperactivity e_L :

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

Taking $s = e_{\text{design}}$ and $t_f = e_{\text{launch}}$, the induced reachability is acyclic and the boundary conditions hold, so $\mathcal{H}$ is a valid PERT/CPM hypergraph.

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

$$H_{\text{und}} = (V, E_H), \quad E_H := \{t(e) \cup h(e) \mid e \in \mathcal{E}\},$$

is an undirected hypergraph in the sense of Definition 2.3.

Proof. For each $e \in \mathcal{E}$, the sets $t(e)$ and $h(e)$ are nonempty subsets of V , hence $t(e) \cup h(e) \subseteq V$ is nonempty. Therefore $E_H \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$, and (V, E_H) is a hypergraph. $\square$

Theorem 7.9 (PERT/CPM graphs embed into PERT/CPM hypergraphs). Every PERT/CPM graph $G = (V, E, d, s, t)$ (Definition 7.1) induces a PERT/CPM hypergraph $\mathcal{H} = (V, \mathcal{E}, t_{\mathcal{H}}, h_{\mathcal{H}}, d_{\mathcal{H}}, s, t)$ 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. Each $(u, v) \in E$ yields a hyperactivity $e_{u,v}$ with singleton predecessor and successor sets. Acyclicity and the boundary conditions carry over from G , and durations agree by definition. Thus $\mathcal{H}$ is a PERT/CPM hypergraph whose singleton-tail/singleton-head specialization reproduces G . $\square$

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

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

A PERT/CPM n -superhypergraph is a tuple

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

where

- $V_n \subseteq \mathcal{P}^n(V_0)$ is a set of n -supervertices containing distinguished *start* and *finish* supervertices $s_n, t_n \in V_n$;
- $E_n \subseteq \mathcal{P}(V_n) \setminus \{\emptyset\}$ is a set of n -superactivities;
- $t, h : E_n \rightarrow \mathcal{P}(V_n) \setminus \{\emptyset\}$ assign to each $e \in E_n$ nonempty predecessor and successor sets $t(e), h(e) \subseteq V_n$;
- $d : E_n \rightarrow \mathbb{R}_{>0}$ assigns positive durations.

These data are required to satisfy:

- (Acyclicity) the reachability relation induced by (t, h) on V_n is acyclic (equivalently, a partial order);
- (Boundary) s_n has no predecessors and t_n has no successors;
- (Connectivity) every $v \in V_n$ lies on some directed hyperpath from s_n to t_n .

Figure 5 summarizes the structure of this example.

Example 7.11 (Software milestones as a PERT/CPM 2-superhypergraph). Let

$$V_0 = \{p = \text{planning_done}, i = \text{implementation_done}, q = \text{testing_done}\}.$$

Define 1-supervertices (phases)

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

At level 2, set

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

Define superactivities $e_1, e_2 \in \mathcal{P}(V_2) \setminus \{\emptyset\}$ by

$$t(e_1) = \{M_1\}, \quad h(e_1) = \{\{Q\}\}, \quad d(e_1) = 5, \quad t(e_2) = \{M_1, \{Q\}\}, \quad h(e_2) = \{M_2\}, \quad d(e_2) = 2.$$

Then e_1 encodes that testing depends on planning and implementation, and e_2 encodes the final milestone depending jointly on all phases.

Theorem 7.12 (Underlying n -superhypergraph). If $\mathcal{H}^{(n)} = (V_n, E_n, t, h, d, s_n, t_n)$ is a PERT/CPM n -superhypergraph, then (V_n, E_n) is an n -superhypergraph in the sense of Definition 2.5.

Proof. By Definition 7.10, $V_n \subseteq \mathcal{P}^n(V_0)$ and $E_n \subseteq \mathcal{P}(V_n) \setminus \{\emptyset\}$. Forgetting t, h, d therefore yields the required n -superhypergraph structure on (V_n, E_n) . $\square$

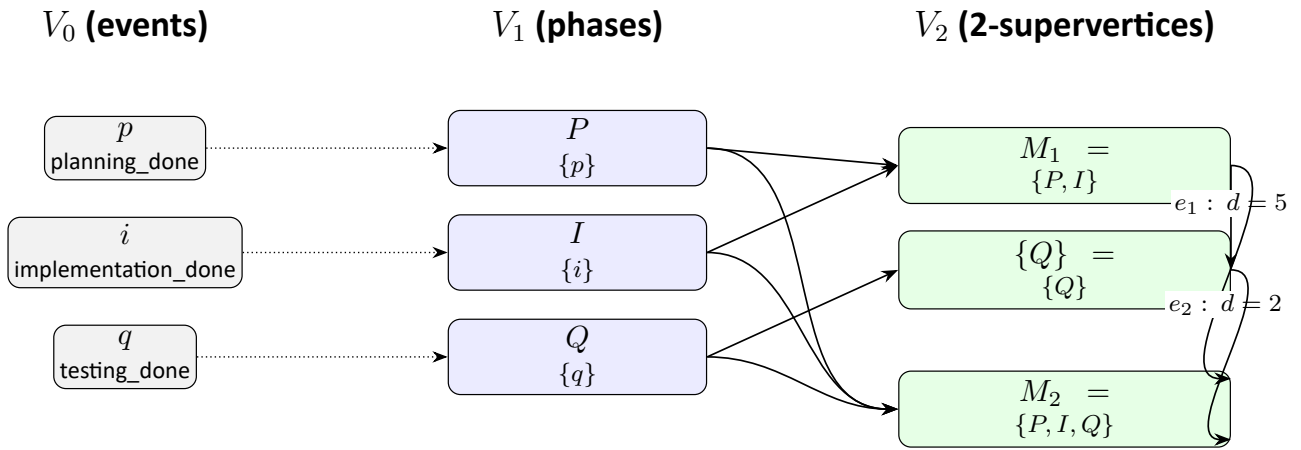


Fig. 5. Software milestones encoded as a PERT/CPM 2-superhypergraph (Example 7.11). Dotted arrows indicate membership $V_0 \rightarrow V_1$; solid arrows indicate containment/lifting and the level-2 superactivities e_1, e_2 with durations.

Theorem 7.13 (PERT/CPM hypergraphs embed as PERT/CPM 1-superhypergraphs). *Let*

$$H = (V_0, \mathcal{E}, t_H, h_H, d_H, s, t_f)$$

be a PERT/CPM hypergraph (Definition 7.6). Let $\iota : V_0 \rightarrow \mathcal{P}(V_0)$ be the singleton injection $\iota(v) = \{v\}$. Set

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

For each $e \in E_1$, define

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

Then $\mathcal{H}^{(1)} = (V_1, E_1, t, h, d, \{s\}, \{t_f\})$ is a PERT/CPM 1-superhypergraph representing H .

Proof. Because $t_H(e)$ and $h_H(e)$ are nonempty, the transported predecessor and successor sets are nonempty subsets of V_1 , so $\mathcal{H}^{(1)}$ satisfies Definition 7.10 with $n = 1$. Acyclicity and the boundary/connectivity conditions are preserved under the singleton injection, since directed hyperpaths correspond bijectively. $\square$

8. Conclusion

This study has presented a unified framework for extending five foundational graph models—factor graphs, signal flow graphs, control flow graphs, dataflow graphs, and PERT/CPM graphs—to hypergraphs and superhypergraphs. The proposed constructions represent multi-way interactions and hierarchical dependencies while retaining the classical graph models as special cases. Formal definitions and illustrative examples demonstrate how these extensions can support dependency modeling across inference, dynamical systems, program analysis, streaming computation, and project scheduling. The results provide a theoretical foundation for further investigation, although their practical applicability requires computational and empirical validation.

Future research will focus on incorporating uncertainty into the proposed frameworks through fuzzy sets [36], neutrosophic sets [37, 38], and plithogenic sets [39]. These extensions could enable more flexible representations of dependencies involving imprecise, indeterminate, or multi-valued information. Further work should also examine computational methods and practical implementations to assess the usefulness of the proposed structures in engineering and intelligent computing applications.

Acknowledgement

This research was not funded by any grant.

Conflicts of Interest

The authors declare no conflicts of interest.

Use of Generative AI and AI-Assisted Tools

Generative AI and AI-assisted tools were used only in a limited supporting role, such as checking English grammar and improving language clarity. They were not used in any ethically inappropriate manner.

References

- [1] Feng, Y., You, H., Zhang, Z., Ji, R., & Gao, Y. (2019). Hypergraph neural networks. *Proceedings of the AAAI conference on artificial intelligence*, 33(01), 3558–3565. <https://arxiv.org/abs/1809.09401>
- [2] Cai, D., Song, M., Sun, C., Zhang, B., Hong, S., & Li, H. (2022). Hypergraph structure learning for hypergraph neural networks. *IJCAI*, 1923–1929. <https://doi.org/10.24963/ijcai.2022/267>.
- [3] Karypis, G. (2003). Multilevel hypergraph partitioning. In *Multilevel optimization in vlsicad* (pp. 125–154). Springer. https://doi.org/10.1007/978-1-4757-3748-6_3.
- [4] Fujita, T., Das, A. K., Das, S., & Mondal, S. P. (2026). Anti-fuzzy hypergraphs and superhypergraphs: Max-oriented uncertainty models for higher-order and hierarchical networks. *Sustainable Machine Intelligence Journal*, 14(3). <https://doi.org/10.63689/3005-3617.1095>.
- [5] Shareef, M. P., Jose, B. R., & Mathew, J. (2026). Llm-semantic enrichment of neutrosophic fuzzy hypergraphs for indeterminacy-driven reinforcement learning and explainable path-based recommendations. *IEEE Access*. <https://doi.org/10.1109/ACCESS.2026.3715708>.
- [6] Shareef, M. P., Jose, B. R., Mathew, J., & PB, R. (2025). Indeterminacy-driven trade-off in reinforcement learning on neutrosophic fuzzy hypergraphs for explainable item recommendation with path-compliant rewards. *IEEE Transactions on Emerging Topics in Computational Intelligence*. <https://doi.org/10.1109/TETCI.2025.3616051>.
- [7] Smarandache, F. (2022). *Introduction to the n-superhypergraph-the most general form of graph today*. Infinite Study. <https://fs.unm.edu/NSS/n-SuperHyperGraph.pdf>
- [8] Hamidi, M., Smarandache, F., & Davneshvar, E. (2022). Spectrum of superhypergraphs via flows. *Journal of Mathematics*, 2022(1), 9158912. <https://doi.org/10.1155/2022/9158912>.
- [9] Ghods, M., Rostami, Z., & Smarandache, F. (2022). Introduction to neutrosophic restricted superhypergraphs and neutrosophic restricted superhypertrees and several of their properties. *Neutrosophic Sets and Systems*, 50, 480–487.
- [10] Hamidi, M., & Taghinezhad, M. (2023). *Application of superhypergraphs-based domination number in real world*. Infinite Study. <https://doi.org/10.22103/jmmr.2023.21203.1415>.
- [11] Fujita, T., & Smarandache, F. (2026a). Foundations of fuzzy, neutrosophic, plithogenic, and uncertain superhypergraphs: A unified framework for modeling hierarchical and uncertainty-aware graph structures in decision-making and real-world applications. *International Journal of Research in Industrial Engineering*. <https://doi.org/10.22105/riej.2026.549395.1695>.
- [12] Diestel, R. (2024). *Graph theory*. Springer (print edition); Reinhard Diestel (eBooks).
- [13] Bretto, A. (2013). Hypergraph theory. *An introduction*. *Mathematical Engineering*. Cham: Springer, 1. <https://doi.org/10.1007/978-3-319-00080-0>.
- [14] Smarandache, F. (2020). *Extension of hypergraph to n-superhypergraph and to plithogenic n-superhypergraph, and extension of hyperalgebra to n-ary (classical-/neutro-/anti-) hyperalgebra*. Infinite Study. <https://fs.unm.edu/NSS/n-SuperHyperGraph-n-HyperAlgebra.pdf>

- [15] Schwartz, I., Yu, S., Hazan, T., & Schwing, A. G. (2019). Factor graph attention. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2039–2048. <https://experts.illinois.edu/en/publications/factor-graph-attention/>
- [16] Dellaert, F. (2021). Factor graphs: Exploiting structure in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1), 141–166. <https://doi.org/10.1146/annurev-control-061520-010504>.
- [17] Frey, B. J., Kschischang, F. R., Loeliger, H.-A., & Wiberg, N. (1997). Factor graphs and algorithms. *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, 35, 666–680. <https://doi.org/10.1109/ITNW.1999.814338>.
- [18] Veerachary, M. (2004). General rules for signal flow graph modeling and analysis of dc-dc converters. *IEEE transactions on Aerospace and Electronic Systems*, 40(1), 259–271. <https://doi.org/10.1109/TAES.2004.1292158>.
- [19] Huggins, W. H. (2007). Signal-flow graphs and random signals. *Proceedings of the IRE*, 45(1), 74–86. <https://doi.org/10.1109/JRPROC.1957.278301>.
- [20] Devkota, S. (2021). *Visualizing control flow graphs* [Doctoral dissertation, The University of Arizona]. <https://www.proquest.com/openview/9f155dec6041a24b75b071c087056baa/1?pq-origsite=gscholar&cbl=18750&diss=y>
- [21] Arlt, S., Rümmer, P., & Schäfer, M. (2013). A theory for control-flow graph exploration. *Automated Technology for Verification and Analysis: 11th International Symposium, ATVA 2013, Hanoi, Vietnam, October 15-18, 2013. Proceedings*, 506–515. https://doi.org/10.1007/978-3-319-02444-8_44.
- [22] Bhattacharyya, S. S., Deprettere, E. F., & Theelen, B. D. (2013). Dynamic dataflow graphs. *Handbook of Signal Processing Systems*, 905–944. https://doi.org/10.1007/978-3-319-91734-4_32.
- [23] Lee, E. A. (1991). Consistency in dataflow graphs. *IEEE Transactions on Parallel and Distributed systems*, 2(2), 223–235. <https://doi.org/10.1109/71.89067>.
- [24] Schneider, K., & Bhagyanath, A. (2023). Consistency constraints for mapping dataflow graphs to hybrid dataflow/von neumann architectures. *ACM Transactions on Embedded Computing Systems*, 22(5), 1–25. <https://doi.org/10.1145/3607869>.
- [25] Mazlum, M., & Güneri, A. F. (2015). Cpm, pert and project management with fuzzy logic technique and implementation on a business. *Procedia-Social and Behavioral Sciences*, 210, 348–357. <https://doi.org/10.1016/j.sbspro.2015.11.378>.
- [26] Gao, Y., Zhang, Z., Lin, H., Zhao, X., Du, S., & Zou, C. (2020). Hypergraph learning: Methods and practices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5), 2548–2566. <https://doi.org/10.1109/TPAMI.2020.3039374>.
- [27] Valencia, E. M. C., Vásquez, J. P. C., & Lois, F. Á. B. (2025). 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. https://digitalrepository.unm.edu/nss_journal/vol84/iss1/28/
- [28] Hamidi, M., Smarandache, F., & Taghinezhad, M. (2023). *Decision making based on valued fuzzy superhypergraphs*. Infinite Study. <https://fs.unm.edu/ScArt/DecisionMakingBasedSuperhypergraphs.pdf>
- [29] Marcos, B. V. S., Willner, M. F., Rosa, B. V. C., Yissel, F. F. R. M., Roberto, E. R., Puma, L. D. B., & Fernández, D. M. M. (2025). Using plithogenic n-superhypergraphs to assess the degree of relationship between information skills and digital competencies. *Neutrosophic Sets and Systems*, 84, 513–524. <https://doi.org/10.5281/zenodo.15641273>.
- [30] Zhu, S. (2025). Neutrosophic n-superhypernetwork: A new approach for evaluating short video communication effectiveness in media convergence. *Neutrosophic Sets and Systems*, 85, 1004–1017. <https://fs.unm.edu/NSS/58ShortVideo.pdf>

-
- [31] Berge, C. (1984). *Hypergraphs: Combinatorics of finite sets* (Vol. 45). Elsevier.
 - [32] Smarandache, F. (2023). *Superhyperfunction, superhyperstructure, neutrosophic superhyperfunction and neutrosophic superhyperstructure: Current understanding and future directions*. Infinite Study. <https://doi.org/10.61356/j.nswa.2023.115>.
 - [33] Das, A. K., Das, R., Das, S., Debnath, B. K., Granados, C., Shil, B., & Das, R. (2025). A comprehensive study of neutrosophic superhyper bci-semigroups and their algebraic significance. *Transactions on Fuzzy Sets and Systems*, 8(2), 80. <https://doi.org/10.71602/tfss.2025.1198050>.
 - [34] Fujita, T., & Smarandache, F. (2026b). *Hypergraph and superhypergraph theory with applications*. Neutrosophic Science International Association (NSIA) Publishing House.
 - [35] Theiling, H. (2002). Control flow graphs for real-time systems analysis. *Universität des Saarlandes, Diss.*
 - [36] Zadeh, L. A. (1965). Fuzzy sets. *Information and control*, 8(3), 338–353. [https://doi.org/10.1016/S0019-9958\(65\)90241-X](https://doi.org/10.1016/S0019-9958(65)90241-X).
 - [37] Broumi, S., Talea, M., Bakali, A., & Smarandache, F. (2016a). Single valued neutrosophic graphs. *Journal of New theory*, (10), 86–101.
 - [38] Broumi, S., Talea, M., Bakali, A., & Smarandache, F. (2016b). Interval valued neutrosophic graphs. *Critical Review, XII, 2016*, 5–33. <https://fs.unm.edu/IntervalValuedNeutrosophicGraphs-CR12.pdf>
 - [39] Smarandache, F. (2018). Plithogeny, plithogenic set, logic, probability, and statistics. *arXiv preprint arXiv:1808.03948*.