Heterogeneous Graph Neural Networks for Assumption-Based Argumentation

Preesha Gehlot[1*], Anna Rapberger[1,2*], Fabrizio Russo[1*], Francesca Toni[1]

[1]Imperial College London
[2]Technical University Dortmund
{a.rapberger, fabrizio}@imperial.ac.uk

Abstract

Assumption-Based Argumentation (ABA) is a powerful structured argumentation formalism, but exact computation of extensions under stable semantics is intractable for large frameworks. We present the first Graph Neural Network (GNN) approach to approximate credulous acceptance in ABA. To leverage GNNs, we model ABA frameworks via a dependency graph representation encoding assumptions, claims and rules as nodes, with heterogeneous edge labels distinguishing support, derive and attack relations. We propose two GNN architectures—ABAGCN and ABAGAT—that stack residual heterogeneous convolution or attention layers, respectively, to learn node embeddings. Our models are trained on the ICCMA 2023 benchmark, augmented with synthetic ABAFs, with hyperparameters optimised via Bayesian search. Empirically, both ABAGCN and ABAGAT outperform a state-of-the-art GNN baseline that we adapt from the abstract argumentation literature, achieving a node-level F1 score of up to 0.71 on the ICCMA instances. Finally, we develop a sound polynomial time extension-reconstruction algorithm driven by our predictor: it reconstructs stable extensions with F1 above 0.85 on small ABAFs and maintains an F1 of about 0.58 on large frameworks. Our work opens new avenues for scalable approximate reasoning in structured argumentation.

1 Introduction

Computational argumentation provides formal tools for modelling defeasible reasoning over conflicting information. In the *abstract* setting, arguments are treated as opaque entities and asymmetric conflicts between them, so-called attacks, as a binary relation, giving rise to abstract argumentation frameworks (AFs) (Dung 1995), which can also be understood as graphs (with arguments as nodes and attacks as edges). By contrast, *structured* formalisms expose the internal composition of arguments. *Assumption-Based Argumentation (ABA)* (Bondarenko et al. 1997) is a prominent structured framework, enabling reasoning in domains such as decision support (Čyras et al. 2021), planning (Fan 2018), and causal discovery (Russo, Rapberger, and Toni 2024).

The building blocks of ABA are *assumptions*, which are the defeasible elements of an ABA framework (ABAF), and *inference rules*, which are used to construct *arguments* based on *assumptions*. *Conflicts* arise between assumption sets S and T if the conclusion of an argument constructed from S is the *contrary* of some assumption in T; in this case we say that S attacks T.

Argumentation semantics provide criteria which render sets of assumptions (so-called *extensions*) jointly acceptable. One of the most popular semantics is the semantics of *stable extensions* which accepts a set of assumptions S only if it has no internal conflicts and attacks every $\{x\}$ with x an assumption not in S. While stable semantics neatly characterises acceptable assumption sets, they are hard to compute: verifying *credulous acceptance* w.r.t. stable extension semantics, i.e., checking if an assumption is contained in any stable extension, is NP-complete (Dimopoulos, Nebel, and Toni 2002). Credulous acceptance plays an important role in several settings, e.g., ABA learning (De Angelis, Proietti, and Toni 2024) relies on it to learn ABAFs from data. Since these settings require fast solutions and real-world ABA frameworks can be very large (Russo, Rapberger, and Toni 2024), relying on exact approaches is often not viable.

In this work, we address this issue by using *Graph Neural Networks (GNNs)* (Scarselli et al. 2009) to predict credulous acceptance. GNNs leverage the relationships between nodes and edges in a graph to learn representations that capture its structure and allow to predict labels for nodes, edges or the whole graph. Their specialised architecture allows GNNs to *amortise* the graph analysis and use its stored weights to predict feature of interest (e.g. node labels, as in our case) in constant time after the initial training, thus introducing significant efficiency at inference time. While GNNs have been successfully used to approximate the acceptability of AFs seen as graphs, e.g., as in (Kuhlmann and Thimm 2019; Malmqvist et al. 2020; Cibier and Mailly 2024), to the best of our knowledge their potential remains unexplored for ABA and any other structured argumentation formalism to date. To make GNNs applicable for ABA, we represent ABAFs via *dependency graphs*. In summary, our contributions are as follows:

- We introduce a faithful dependency graph encoding of ABAFs that distinguishes assumption, claim and rule nodes as well as support, derive, and attack edges.
- We develop two heterogeneous GNN architectures—ABAGCN and ABAGAT—composed of residual stacks

*These authors contributed equally.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

of relation-specific convolutional or attention layers, respectively, enriched with learnable embeddings and degree-based features.

- We implement a training and evaluation pipeline for credulous acceptance combining 380 ICCMA 2023 ABA benchmarks[1] with 19,500 additional, synthetically generated ABAFs. Our experiments demonstrate that both ABAGCN and ABAGAT outperform an AF-based GNN baseline based on mapping ABAFs onto AFs and then applying state-of-the-art AFGCNv2, achieving node-level F1 up to 0.71 on small ABAFs and 0.65 overall.
- Based on our credulous acceptance predictor, we develop a sound poly-time extension-reconstruction algorithm which computes stable extensions with high fidelity. Our algorithm attains F1 of 0.85 on small frameworks and remains above 0.58 on large ABAFs.

Trained models, code to reproduce data and experiments are provided at https://anonymous.4open.science/r/GNN4ABA.

2 Related Work

Kuhlmann and Thimm (2019) were the first to frame credulous acceptance in an AF as a graph-classification problem. They generated random AFs, labeled each argument's acceptability using the exact CoQuiAAS solver (Lagniez, Lonca, and Mailly 2015), and trained a Graph Convolutional Network (GCN) on two feature sets: the raw adjacency matrix and adjacency augmented with each node's in- and out-degree. Including degree information improved accuracy to $\sim 80\%$, with F1 scores below 0.4, and class-balancing during training enhanced detection of under-represented arguments. Despite only marginal effects from graph topology or dataset size, their GCN reduced runtime from over an hour (CoQuiAAS) to under 0.5s for a full test set.

Malmqvist et al. (2020) then proposed AFGCN, which augments node features with 64-dimensional DeepWalk embeddings (Perozzi, Al-Rfou, and Skiena 2014) before feeding them—alongside the AF's adjacency matrix—through 4–6 GCN–dropout blocks with residual connections. To address the skewed accept/reject ratio in real AFs, they exclude AFs with very few accepted arguments and introduce a randomised per-epoch masking of labels, forcing the model to infer hidden acceptabilities. This scheme, rather than network depth or explicit class-balancing, accounts for most of AFGCN's increase to $\sim 82\%$ accuracy on rebalanced data (62% reported for (Kuhlmann and Thimm 2019) on the same data). Malmqvist, Yuan, and Nightingale (2024) enrich argument representations with graph-level metrics (centralities, PageRank, colouring) and use four GCN–ReLU–dropout blocks plus an epoch-wise reshuffle-and-rebalance routine and a pre-check on the grounded extension (Dung 1995) for faster and more accurate inference, proposing AFGCNv2.

Cibier and Mailly (2024) optimise AFGCNv2 by reimplementing AF parsing and graph-metric computation in Rust, slashing preprocessing time and memory usage. They evaluate AFGCNv2 against two variants—adding five semantic features to random features, or retaining only 11 "meaningful" features—and assess each with and without dropout. Their results show that pruning degrades accuracy, and that the no-dropout version of "feature-enhanced" version achieves the best overall performance ($\sim 83\%$ accuracy vs $\sim 75\%$ reported for AFGCNv2 on the same data), highlighting the importance of rich semantic features and careful dropout placement. Finally, they introduce AFGAT, a three-layer GATv2 (an updated attention mechanism (Brody, Alon, and Yahav 2022) with multi-head attention, see Section 3.2), which surpasses all GCN variants obtaining $\sim 87\%$ on the same benchmark data from ICCMA.

Craandijk and Bex (2020) propose AGNN, which learns argument embeddings through iterative message passing that internalises conflict-freeness and defense, then uses the resulting acceptance probabilities to drive a constructive backtracking algorithm for extension enumeration. We build on this enumeration strategy for our proposed extension-reconstruction algorithms in Section 6, adapting it to the nuances of ABAFs instead of AFs.

We adopt AFGCNv2 (Malmqvist, Yuan, and Nightingale 2024) (with AFs instantiated with ABAFs) as our baseline[2] but, inspired by the results in (Cibier and Mailly 2024), we replace its handcrafted features with learnable embeddings and experiment with both GCNs and GAT layers. Additionally, we handle class imbalance via loss weighting rather than mini-batch rebalancing and disable the grounded-extension heuristic to test the models' predictive accuracy on the entire frameworks rather than only on the non-grounded portion of it. In contrast to the AF-focused solvers discussed, we present the first native approximate solver for ABAFs, combining an enriched graphical representation with heterogeneous GCN and GAT-based prediction of assumption acceptability incorporated in an AGNN-inspired extension-reconstruction algorithm.

3 Preliminaries

We recall the relevant elements of abstract and assumption-based argumentation, as well as GNNs.

3.1 Computational Argumentation

An argumentation framework (AF) (Dung 1995) is a directed graph $F = (\mathsf{A}, \mathsf{R})$ where A are arguments and $\mathsf{R} \subseteq \mathsf{A} \times \mathsf{A}$ is an *attack* relation. For $x, y \in \mathsf{A}$, if $(x, y) \in \mathsf{R}$ we say x *attacks* y; for $S, T \subseteq \mathsf{A}$, if $(x, y) \in \mathsf{R}$ for some $x \in A, y \in T$, we say S *attacks* T; E is *conflict-free* ($E \in cf(F)$) iff it does not attack itself. A semantics σ is a function that assigns to each AF a set of sets of arguments, so-called *extensions*. We focus on stable extensions.

Definition 3.1. *Let $F = (\mathsf{A}, \mathsf{R})$ be an AF. A set $E \subseteq \mathsf{A}$ is* stable *($E \in stb(F)$) iff E is conflict-free and attacks each $x \in \mathsf{A} \setminus E$.*

Assumption-based Argumentation We assume a *deductive system*, i.e. a tuple $(\mathcal{L}, \mathcal{R})$, where $\mathcal{L}$ is a set of sentences and $\mathcal{R}$ is a set of inference rules over $\mathcal{L}$. A rule $r \in \mathcal{R}$ has

[1] Available at https://zenodo.org/records/8348039

[2] Available at https://github.com/lmlearning/AFGraphLib/tree/main/AFGCNv2. We could not use the improved versions from (Cibier and Mailly 2024) as no weights were made available.

the form $a_0 \leftarrow a_1, \ldots, a_n$, s.t. $a_i \in \mathcal{L}$ for all $0 \leq i \leq n$; $head(r) := a_0$ is the *head* and $body(r) := \{a_1, \ldots, a_n\}$ is the (possibly empty) *body* of r.

Definition 3.2. *An ABA framework (ABAF) (Bondarenko et al. 1997) is a tuple $(\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$, where $(\mathcal{L}, \mathcal{R})$ is a deductive system, $\mathcal{A} \subseteq \mathcal{L}$ a (non-empty) set of assumptions, and $^- : \mathcal{A} \to \mathcal{L}$ a contrary function.*

An ABAF $\mathcal{D}$ is *flat* if $head(r) \notin \mathcal{A}$ for all $r \in \mathcal{R}$. In this work we focus on *flat* and *finite* ABAFs, i.e. $\mathcal{L}$ and $\mathcal{R}$ are finite. We also restrict attention to $\mathcal{L}$ consisting of atoms.

An atom $p \in \mathcal{L}$ is *tree-derivable* (Dung, Kowalski, and Toni 2009) from sets of assumptions $S \subseteq \mathcal{A}$ and rules $R \subseteq \mathcal{R}$, denoted by $S \vdash_R p$, if there is a finite rooted labeled tree t s.t. i) the root of t is labeled with p, ii) the set of labels for the leaves of t is equal to S or $S \cup \{\top\}$, and iii) for each node v that is not a leaf of t there is a rule $r \in R$ such that v is labeled with $head(r)$ and labels of the children correspond to $body(r)$ or $\top$ if $body(r) = \emptyset$. We write $S \vdash p$ iff there exists $R \subseteq \mathcal{R}$ such that $S \vdash_R p$; $S \vdash p$ is called an *ABA argument*.

Let $S \subseteq \mathcal{A}$. By $\overline{S} := \{\overline{a} \mid a \in S\}$ we denote the set of all contraries of S. S *attacks* $T \subseteq \mathcal{A}$ if there are $S' \subseteq S$ and $a \in T$ s.t. $S' \vdash \overline{a}$; if S attacks $\{a\}$ we say S attacks a. S is conflict-free ($S \in cf(\mathcal{D})$) if it does not attack itself. We recall the definition stable extensions for ABA.

Definition 3.3. *Let $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABAF. A set $S \subseteq \mathcal{A}$ is a* stable extension *($S \in stb(\mathcal{D})$) iff S is conflict-free and attacks each $x \in \mathcal{A} \setminus S$.*

Definition 3.4. *An assumption $a \in \mathcal{A}$ is* credulously accepted *w.r.t. stable extension semantics in an ABAF $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ iff there is some $S \in stb(\mathcal{D})$ with $a \in S$; a is* rejected *if it is not credulously accepted.*

Example 3.5. *Consider an ABAF $(\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ with $\mathcal{L} = \{a, b, c, d, p, \overline{a}, \overline{b}, \overline{c}, \overline{d}\}$, $\mathcal{A} = \{a, b, c, d\}$ with contraries $\overline{a}$, $\overline{b}$, $\overline{c}$, $\overline{d}$, respectively, and rules $r_1, \ldots, r_4$, respectively:*

$$\overline{c} \leftarrow a, d \qquad p \leftarrow b \qquad \overline{d} \leftarrow c \qquad \overline{a} \leftarrow p, c$$

We obtain two stable extensions: $\{b, c\}$ and $\{a, b, d\}$. Indeed, $\{b, c\}$ attacks a and d since $\{b, c\} \vdash \overline{a}$ and $\{c\} \vdash \overline{d}$; and $\{a, b, d\}$ attacks c since $\{a, d\} \vdash \overline{c}$.

Viewing tree derivations as arguments, an ABAF induces an AF as follows (Čyras et al. 2018).

Definition 3.6. *The associated AF $F_{\mathcal{D}} = (A, R)$ of an ABA $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ is given by $A = \{S \vdash p \mid \exists R \subseteq \mathcal{R} : S \vdash_R p\}$ and R such that $(S \vdash p, S' \vdash p') \in R$ iff $p \in \overline{S'}$.*

Semantics of AFs and ABAFs correspond as follows.

Proposition 3.7. *Let $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABAF and $F_{\mathcal{D}}$ its associated AF. If $E \in stb(F_{\mathcal{D}})$ then $\bigcup_{S \vdash p \in E} S \in stb(\mathcal{D})$; if $S \in stb(\mathcal{D})$ then $\{S' \vdash p \mid \exists S' \subseteq S : S' \vdash p\} \in stb(F_{\mathcal{D}})$.*

Note that the AF instantiation can be exponential in the size of the given ABAF $\mathcal{D}$. To tackle this issue, Lehtonen et al. (2023) propose a poly-time preprocessing technique that yields an ABAF so that the resulting AF is polynomial in the size of $\mathcal{D}$. Their procedure handles the derivation circularity and flattens out the nested argument construction; they show that this construction preserves semantics under projection. In Section 6 we use this procedure to construct the AF-based GNN baseline that we use to evaluate the performance of our native ABAF GNNs.

3.2 Graph Neural Networks

A neural network (NN) is a parametrised function that maps an input vector $\mathbf{x} \in \mathbb{R}^n$ to an output through successive layers of neurons. We set the input neurons as $\mathbf{h}^{(0)} = \mathbf{x}$ and, denoting by $\mathbf{h}^{(l)}$ the row-vector of neurons at layer l, each neuron at layer l computes

$$h_i^{(l)} \;=\; f\big((\mathbf{w}_i^{(l-1)})^\top \mathbf{h}^{(l-1)} + b_i^{(l-1)}\big),$$

where $\mathbf{w}_i^{(l)}$ and $b_i^{(l)}$ are the learnable weight-vector and bias for neuron i in layer l, and f is a (possibly non-linear) activation function (Bishop 2007). In supervised learning, the NN is trained by comparing predictions to ground-truth labels via a loss function (e.g. cross-entropy), and parameters are optimised by back-propagation and gradient descent.

Graph neural networks (GNNs) generalise NNs to consume graph-structured data $G = (V, E)$, by jointly leveraging a node-feature matrix $\mathbf{X} \in \mathbb{R}^{|V| \times d}$ and an adjacency matrix $\mathbf{A}$. Through repeated neighbourhood-aggregation steps—each propagating (also known as *message-passing*) and combining information from a node's neighbours—a GNN learns *embeddings* that encode both local topology and node attributes (Bronstein et al. 2017). These embeddings can serve tasks such as node classification, where a final projection and sigmoid, with a threshold (possibly) tuned for class imbalance, yield per-node labels. Popular aggregation schemes include Graph Convolutional Networks (Kipf and Welling 2017, **GCNs**) and Graph Attention Networks (Velickovic et al. 2018, **GATs**).

GCNs perform matrix-based updates across all nodes simultaneously and treat every neighbour equally. GCNs update node representations by aggregating feature information from each node and its neighbours. At each layer l, the representation matrix $\mathbf{H}^{(l)}$ is computed from the previous layer $\mathbf{H}^{(l-1)}$ and the graph structure as:

$$\mathbf{H}^{(l)} = \sigma\left(\mathbf{\Delta}^{-1/2}\tilde{\mathbf{A}}\mathbf{\Delta}^{-1/2}\mathbf{H}^{(l-1)}\mathbf{W}^{(l-1)}\right),$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix of the graph with added self-loops (so each node includes its own features in the aggregation), $\mathbf{\Delta}$ is the diagonal degree matrix of $\tilde{\mathbf{A}}$, $\mathbf{W}^{(l)}$ is a learnable weight matrix for layer l, and σ is a (possibly non-linear) activation function. The inclusion of self-loops ensures that nodes can retain and refine their own features across layers, rather than relying solely on neighbouring information. Typically, $\mathbf{H}^{(0)} = \mathbf{X}$.

GATs compute learned attention coefficients (Vaswani et al. 2017) to weight each neighbour's contribution differently. Unlike GCNs, which treat all neighbouring nodes equally during aggregation, GATs introduce learnable attention mechanisms to assign different weights to different neighbours. Specifically, for a node i, attention coefficients

α_{ij} are computed between i and each of its neighbours j using shared attention:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_i \,\|\, \mathbf{W}\mathbf{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}(i)} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_i \,\|\, \mathbf{W}\mathbf{h}_k]\right)\right)},$$

where $\mathbf{W}$ is a learnable weight matrix, $\mathbf{a}$ is a learnable attention vector, $[\cdot \,\|\, \cdot]$ denotes vector concatenation, and $\mathcal{N}(i)$ is the set of neighbours of node i. LeakyReLU is a non-linear activation function that allows a small negative slope (typically $\epsilon = 0.2$) for negative inputs, helping to avoid dead neurons and improve gradient flow. The updated representation of node i is then computed as a weighted sum of the transformed features of its neighbours: $\mathbf{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}\mathbf{W}\mathbf{h}_j\right)$. To stabilise the learning process, GATv2 (Brody, Alon, and Yahav 2022) employs multi-head attention: multiple independent attention mechanisms are applied in parallel, and their outputs are concatenated (in intermediate layers) or averaged (in the output layer).

When modelling heterogeneous graphs—where edges carry distinct semantics—one typically employs relation-specific transformations, e.g. via a Heterogeneous Graph Convolution (HGC) module (Wang et al. 2019), and adds residual connections and normalisation between layers to stabilise training. To prevent overfitting, dropout (Srivastava et al. 2014) can be employed after each layer, randomly setting to zero a proportion of the weights, and early stopping to terminate training when the validation loss does not improve for a number of steps greater than a patience parameter.

4 Predicting Acceptance with GNNs

Here we detail our proposed GNN architecture to predict credulous acceptance of assumptions in an ABA framework. The core and novel component is the *dependency graph*, a faithful graph-based representation of an ABAF, that guides the GNN learning of the ABA features that help predicting credulous acceptance (Section 4.1). The neural machinery adopted to learn to classify the assumption nodes of the dependency graph as *credulously accepted* or *rejected* w.r.t. stable extension semantics is then detailed in Section 4.2.

4.1 Dependency Graph

Our proposed graph representation comprises nodes for each atom type (assumption and non-assumption) and for each rule, and three edge types: *support* edges ($+$) linking body atoms to rule nodes; *derive* edges ($\triangleright$) linking rule nodes to their head elements; and *attack* edges ($-$) connecting contraries to their assumption nodes.

Definition 4.1. *Let $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^-)$ be an ABAF. The* dependency graph *$G_D = (V, E, l)$ of D is an edge-labelled graph with nodes $V = \mathcal{L} \cup \mathcal{R}$, edges*

$$E = \{(p, r) \mid r \in \mathcal{R}, p \in \mathit{body}(r)\} \cup \{(r, p) \mid r \in \mathcal{R}, p \in \mathit{head}(r)\} \cup \{(p, a) \mid a \in \mathcal{A}, \overline{a} = p\},$$

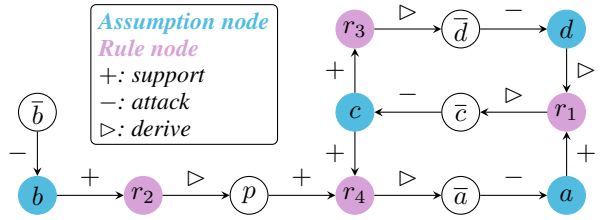
and edge labellings, for $e \in E$,

$$l(e) = \begin{cases} +, & \text{if } e = (p, r), r \in \mathcal{R}, p \in \mathit{body}(r) \\ \triangleright, & \text{if } e = (r, p), r \in \mathcal{R}, p \in \mathit{head}(r) \\ -, & \text{if } e = (p, a), a \in \mathcal{A}, \overline{a} = p \end{cases}$$

Observation 4.2. *$G_\mathcal{D} \neq G_{\mathcal{D}'}$ for every two ABAFs $\mathcal{D}, \mathcal{D}'$, $\mathcal{D} \neq \mathcal{D}'$, i.e., $G_\mathcal{D}$ is unique for each ABAF $\mathcal{D}$.*

Since $G_\mathcal{D}$ contains information about all assumptions, contraries, and rules of a given ABAF $\mathcal{D}$, the ABAF can be fully recovered from $G_\mathcal{D}$, thus the semantics are preserved, as for the AF instantiation. In contrast to the AF instantiation, the construction of the dependency graph requires a single pass over the atoms and rules of $\mathcal{D}$.

Example 4.3. *The dependency graph $G_\mathcal{D}$ of the ABAF $\mathcal{D}$ from Example 3.5 is as follows:*



We can use $G_\mathcal{D}$ to check that b and c jointly derive $\overline{a}$, using rules r_2 and r_4.

Rapberger, Ulbricht, and Wallner (2022) define a dependency graph for ABA in the spirit of dependency graphs for logic programming (Konczak, Linke, and Schaub 2006; Fandinno and Lifschitz 2023). However, their representation does not fully preserve the structure of the ABAF, preventing the possibility to extract the ABAF's extensions from a given dependency graph.[3] In contrast, our dependency graph is unique for every ABAF and preserves the semantics.

4.2 Neural Architecture

Our GNN operates on the dependency graph's adjacency matrix $\mathbf{W} \in \{0, 1\}^{|V|\times|V|}$ and an initial node-feature matrix $\mathbf{F} \in \mathbb{R}^{|V|\times 2}$, where each row F_i contains the in- and out-degrees of node i for each node type: assumptions $\mathcal{A}$, non-assumptions (or claims) $\mathcal{C} = \mathcal{L} \setminus \mathcal{A}$, and rules $\mathcal{R}$. In addition, we maintain a learnable embedding matrix $\mathbf{L} \in \mathbb{R}^{|V|\times d_e}$ with embedding dimension d_e, whose entries are optimised jointly with the rest of the GNN. We show a diagram of our model architecture in Figure 1. The core backbone of the GNN consists of M blocks. In each block m:

- A HGC layer performs relation-specific neighbourhood aggregation on the current embeddings $H^{(m-1)}$, using either GCN or GAT convolutional kernels (see Section 3 for details).
- The aggregated output is passed through functional layers for stability and regularisation: we use layer-normalisation, ReLU activation, and dropout with rate δ.
- A residual connection aggregates the block input $H^{(m-1)}$ to its output, yielding $H^{(m)}$.

[3] In brief, the dependency graph from (Rapberger, Ulbricht, and Wallner 2022) does not include rule nodes; positive edges (p, q) are introduced whenever p is contained in some rule body of a rule with head q. For example, the rule $(\overline{a} \leftarrow p, c)$ from Example 4.3 would yield the same edges as having two rules $(\overline{a} \leftarrow p), (\overline{a} \leftarrow c)$. The introduction of rule nodes was inspired by the conjunction node representation in (Li, Wang, and Gupta 2021).

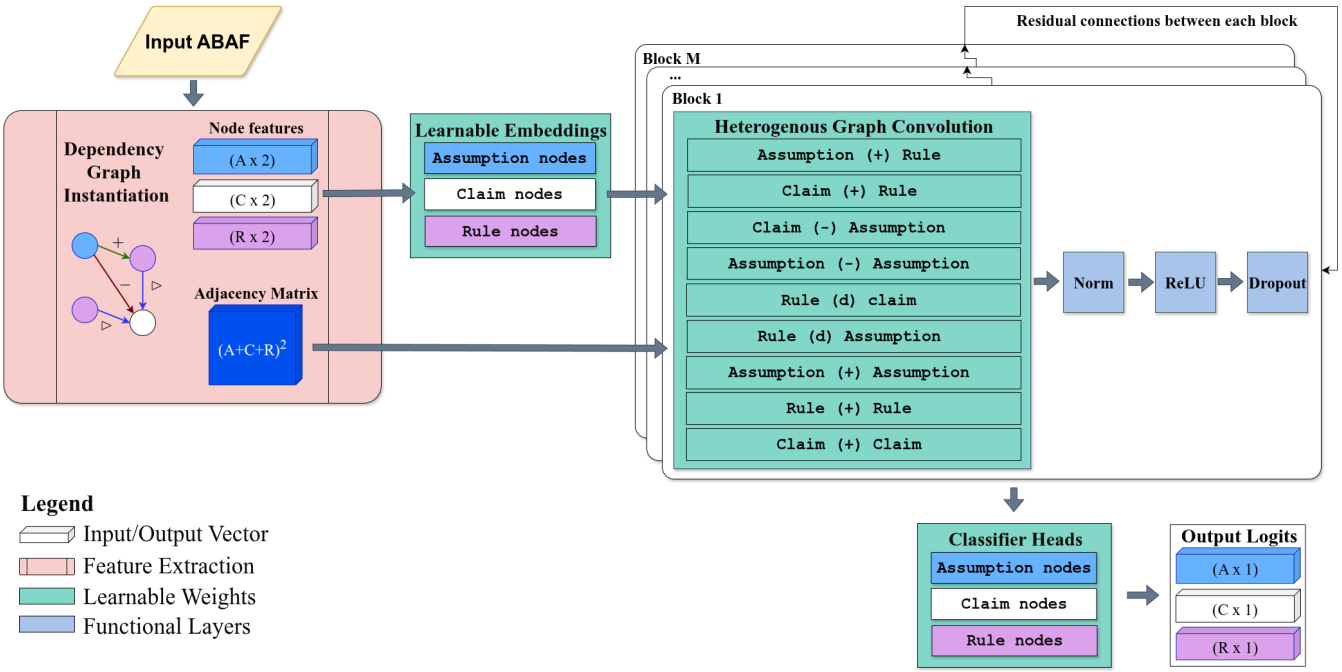


Figure 1: Diagram of the model architecture, including both the feature extraction (using the dependency graph described in Section 4.1) and the GNN described in Section 4.2. The Heterogeneous Graph Convolution (HGC) module would apply Convolutional Layers for ABAGCN, or Attention Layers for ABAGAT.

This design allows distinct transformations per edge type while preserving gradient flow in deep architectures consisting of many layers. Each of these components (HGC and functional layers linked by residual connections) form a so called *block*. After M blocks, with the parameter M tuned on validation data, we extract the rows of $H^{(M)}$ corresponding to assumption, claim and rule nodes and apply a separate linear classifier head to produce logits $z \in \mathbb{R}^{A+C+R}$, with $A = |\mathcal{A}|, C = |\mathcal{C}|$, and $R = |\mathcal{R}|$.

At inference time, a sigmoid activation yields the predicted probability of credulous acceptance for each assumption. Using a threshold τ, again tuned on validation data, we classify assumptions (and potentially also claims and rules) as accepted or rejected through a forward pass of the GNN that process the input features with our optimised weights obtained via supervised training.

We train the GNNs by minimising a weighted binary cross-entropy loss on the assumption logits, using the Adam optimiser with learning rate λ. Class weights compensate for the imbalance between accepted and rejected assumptions (see Section 6.1 and Table 1 in Appendix B.1 for details on the training and test data).

All hyperparameters—including the number of layer blocks M, the embedding dimension d_e, hidden dimension of each convolution layer, the dropout rate δ, the learning rate λ, the batch size, class-weight multiplier, and classification threshold τ—are chosen via Bayesian optimisation (Snoek, Larochelle, and Adams 2012) with 3-fold cross-validation, as implemented in (Biewald 2020). Details of tuning and hyperparameters are in Appendix B.3.

5 Extension Reconstruction

Here we outline our proposed extension-reconstruction algorithm, whose pseudo-code is shown in Algorithm 1. Given an ABAF $\mathcal{D}$ and a predictor $\mathcal{M}$ in input, we pick the highest scoring assumption a^* and mark it as accepted (line 3). Before making our next guess, we modify $\mathcal{D}$ so that the extensions of the modified ABAF $\mathcal{D}'$ (under projection) correspond to the extensions of $\mathcal{D}$ containing a^* (line 4). Since our prediction is imperfect, this modification may have introduced some inconsistencies in $\mathcal{D}'$, which we check in line 5. If the check passes, we add a^* to our extension $\mathcal{E}$.

We outline the $\text{MODIFY}(\mathcal{D}, a^*)$ procedure for a given $\mathcal{D}$ and a^*. When removing an assumption a^* predicted accepted, we aim at ensuring that the extensions of the modified $\mathcal{D}'$ projected onto $\mathcal{A}$ correspond to the extensions of $\mathcal{D}$ that contain a^*. We restrict to $\mathcal{A}$ because we introduce dummy-assumptions to ensure that rules that derive the contrary of a^* act as constraints. Concretely, $\text{MODIFY}(\mathcal{D}, a^*)$ performs the following steps:

1. We remove the assumption a^* and all occurrences of it; that is, we let $\mathcal{A}' = \mathcal{A} \setminus \{a^*\}$ and we replace any rule r with $r' = head(r) \leftarrow body(r) \setminus \{a^*\}$.
2. We modify each rule r with $head(r) = \overline{a^*}$: for each such rule r, we introduce a new dummy assumption d_r and replace r with $r' = \overline{d_r} \leftarrow body(r) \cup d_r$. This modification ensures that not all body elements of r can be true (accepted or derived) at the same time.
3. If a^* is the contrary of another assumption c, then c is rejected. We remove c and any rule r with $c \in body(r)$.

Algorithm 1: Extension reconstruction sketch

Require: ABAF $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$, predictor $\mathcal{M}$
1: Initialize extension $\mathcal{E} \leftarrow \emptyset$
2: **while** $\mathcal{A} \neq \emptyset$ **do**
3: $a^* \leftarrow \arg\max_{a \in \mathcal{A}} P_{\mathcal{M}}(a)$
4: $\mathcal{D}' \leftarrow \text{MODIFY}(\mathcal{D}, a^*)$
5: **if** ISCONFLICTING($\mathcal{D}'$) **then**
6: **break**
7: **else**
8: $\mathcal{E} \leftarrow \mathcal{E} \cup \{a^*\}$
9: **return** $\mathcal{E}$

4. Lastly, we remove all facts from $\mathcal{D}$: for each rule r with $body(r) = \emptyset$, we remove r from $\mathcal{R}$ and $p = head(r)$ from the body of each rule r', i.e., we replace r' with $r'' = head(r') \leftarrow body(r') \setminus \{p\}$. If p is the contrary of an assumption c, the assumption is attacked and we proceed as in Step 3. We repeat this process until there are no remaining facts.

Note that $\text{MODIFY}(\mathcal{D}, a^*)$ runs in polynomial time; naively, the computation requires a loop over all rules in $\mathcal{R}$, for each $p \in \mathcal{L}$; thus, Algorithm 1 is in P. We establish the soundness of Algorithm 1.

Proposition 5.1. *For an ABAF $\mathcal{D}$ and an assumption a^*, let $\mathcal{D}'$ denote the outcome of* $\text{MODIFY}(\mathcal{D}, a^*)$. *If a^* is credulously accepted in $\mathcal{D}$, then the following holds.*

$$\{S \cap \mathcal{A} \mid S \in stb(\mathcal{D}')\} = \{S \setminus \{a^*\} \mid S \in stb(\mathcal{D}), a^* \in S\}$$

The proof of Proposition 5.1 is given in Appendix A. With a perfect predictor of acceptance, Algorithm 1 would return a valid stable extension. With imperfect predictions, if a^* is not accepted in $\mathcal{D}$, Algorithm 1 will encounter a rule of the form $\overline{d_r} \leftarrow d_r$ which indicates a conflict. ISCONFLICTING($\mathcal{D}'$) checks if such a rule exists (line 5). The partially computed set is returned if $\mathcal{A} = \emptyset$ or if a conflict is found.

6 Empirical Evaluation

In this section we assess the effectiveness of our proposed GNN models—both the convolutional (ABAGCN) and attention-based (ABAGAT) variants—against our AFGCNv2-based baseline on two tasks over ABAFs:

1. **Credulous acceptance classification**: predicting, for each assumption, whether it is in any stable extension.
2. **Extension reconstruction**: recovering a full set of accepted assumptions (i.e. a predicted extension).

For task 1 we report node-level precision, recall, F1 and accuracy. For task 2 we treat each stable extension as a set and measure how well our predicted sets match ground truth using extension-level F1 (details in Appendix B.2).

6.1 Data

ICCMA 2023 included an ABA track–but not an approximate ABA one–whose benchmark comprised of 400 ABAFs defined by four parameters: number of atoms, assumption proportion, maximum rules per atom, and maximum rule-body length. Ground-truth labels are obtained by running ASPForABA[4] (Lehtonen et al. 2024) on every test instance with timeout threshold 10 min, yielding 380 benchmark ABAFs with exact annotations within the ICCMA data.

To obtain a larger training corpus, we used the ABAF generator from (Lehtonen et al. 2024) to produce 48,000 flat ABAFs (47,794 after timeout filtering when producing labels), extending the parameter ranges. Parameters are shown in Table 1 in Appendix B.1. To match the original assumptions acceptance rate (36.7% overall, 60.3% within ABAFs with at least one stable extension), we curated a subset of 19,500 ABAFs (13,000 with at least one accepted assumption, 6,500 with none), yielding 32.5% overall acceptance and 53.4% within ABAFs with at least a solution.

Finally, we used a 75:25 train–test split, stratified across three groups—small ICCMA (25–100 atoms), full ICCMA, and our generated dataset—to ensure a fair comparison with AFGCNv2, which cannot process ABAFs larger than 100 atoms. Accordingly, all results are reported on three test sets: ICCMA small, ICCMA full and ICCMA+Generated.

6.2 Baseline

Our baseline implements a three-stage pipeline that reduces ABAF credulous acceptance to an AF classification task:

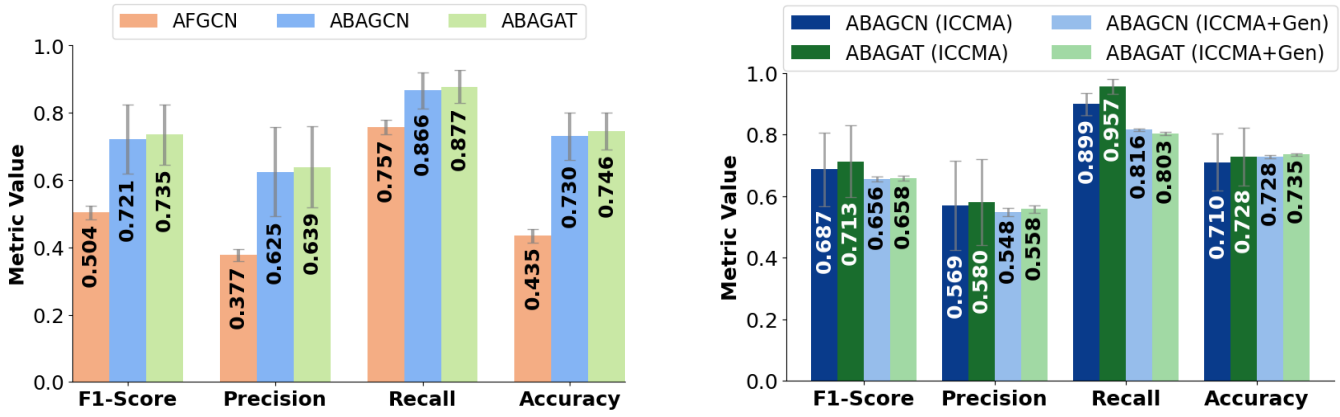
1. **ABAF-AF translation:** Given an input ABAF, we use the method from (Lehtonen et al. 2023) and implemented in the AcBar toolkit[5] (Lehtonen et al. 2021) to convert the input ABAF into a (poly-sized) AF in polynomial time.
2. **Argument classification via AFGCNv2:** The generated AF is represented by its adjacency matrix and graph-level features, then fed into the AFGCNv2 solver (see Section 2) using pre-trained weights for credulous acceptance under stable semantics. To ensure a fair comparison with our models, we disable its grounded-extension pre-check and apply a classification threshold—tuned on a held-out validation set—to the sigmoid output so that any argument with score above the threshold is accepted.
3. **Element acceptability recovery:** Finally, we derive assumption acceptance in the original ABAF: an assumption a is accepted if the argument $\{a\} \vdash a$ is accepted.

This baseline leverages an existing AF-based GNN solver, providing a direct comparison for our native ABAF GNNs.

6.3 Credulous Acceptance Classification

We first evaluate how well each model predicts assumption-level acceptance under stable semantics. Figure 2a shows that both our GNNs substantially outperform the AF baseline on the small ICCMA dataset ($|\mathcal{L}| < 100$). We test on this set because AFGCNv2 cannot handle ABAFs greater than this size. F1 increases from 0.5 (AFGCN) to 0.72 (ABAGCN) and 0.74 (ABAGAT); recall climbs by about 10 points; precision improves by about 25 points; and accuracy rises by over 30 points. ABAGAT achieves the highest results across all metrics on this test set, but it is not significantly different from ABAGCN. Statistical tests are provided in Table 3 in Appendix B.5.

[4] Available at https://bitbucket.org/coreo-group/aspforaba.
[5] Available at https://bitbucket.org/lehtonen/acbar



(a) AF vs ABA GNNs on ICCMA data with $|\mathcal{L}| < 100$

(b) CGN vs GAT on ICCMA and generated data

Figure 2: Model comparison according to F1, Precision, Recall and Accuracy on different cuts of data: small ICCMA ABAFs (with less than 100 elements) in panel (a) to be able to compare to the AFGCN baseline; Comparison between the full ICCMA test set and the test set augmented with our generated data in panel (b).

In Figure 2b we examine the effect of testing only on the ICCMA data versus adding our generated instances. Both ABAGCN and ABAGAT see modest gains in accuracy when testing on the generated ABAFs together with the ICCMA ones, but incur a notable drop in precision and recall, resulting in a decrease in F1. This pattern indicates that synthetic augmentation possibly produced more challenging ABAFs than the ones in the competition. Crucially, despite this worsening in performance, both GNN variants maintain strong performance across the held-out ICCMA and generated test split, demonstrating robust generalisation to larger and more diverse ABA frameworks. Here ABAGCN and ABAGAT do not show significant difference across metrics, apart from ABAGAT surpassing ABAGCN in accuracy and recall on the ICCMA test, while ABAGCN showing significantly better recall than ABAGAT on the ICCMA+Gen set.

Comparing the small ICCMA results in Figure 2a to the (full) ICCMA bars in Figure 2b, F1 dips by under one percentage point, precision by roughly two points and accuracy by about one point on the full dataset, while recall actually climbs by two, to six points, for ABAGCN and ABAGAT, respectively. In other words, on larger ABAFs both GNNs trade a bit of overall and positive-class fidelity for stronger coverage of accepted assumptions.

6.4 Extension Reconstruction

For our second experiment, we test the poly-time algorithm from Section 5, using our approximate ABA models to calculate a stable extension. Our constructive extension-reconstruction method achieves very high F1 (0.85) on small ABAFs (10 atoms), but performance drops sharply to around 0.60 by 50–100 atoms and then plateaus at approximately 0.58 for larger frameworks (see Figure 3 in Appendix B.4). This decline reflects how early misclassifications by the GNN propagate through subsequent modification steps, reducing both precision and recall, but highlights opportunities for future work to mitigate degradation.

We also compare runtimes on 15 of the most challenging ABAFs (4,000–5,000 atoms) from our dataset, held out in the test set. The exact ASPForABA solver averaged 435s per instance, whereas our approximate extension reconstruction ran in 192s—2.3× faster—while still achieving F1 of 0.68 (0.77 accuracy). These results highlight the potential of GNN-driven approximate reasoning to yield speed-ups over exact methods, maintaining a similar level predictive performance as in the acceptance classification task.

7 Conclusion

We have demonstrated that heterogeneous GNNs over our dependency-graph encoding can accurately and efficiently approximate credulous acceptance in ABA under stable semantics. Both ABAGCN and ABAGAT outperform an AF-based GNN baseline—achieving node-level F1 of 0.65 overall and up to 0.71 on small frameworks—while our poly-time extension-reconstruction procedure reconstructs stable sets with F1 greater than 0.85 on small ABAFs and maintains F1 of about 0.58 at scales of 1,000 atoms. Crucially, on the 4,000–5,000-atom ABAFs where ASPForABA average runtime is 435s, our approximate extension are derived in 192s (2.3× faster) with F1 of 0.68, underscoring substantial runtime gains without sacrificing predictive quality. These findings show that GNNs can bridge the gap between accuracy and tractability in structured argumentation.

Future work includes integrating lightweight symbolic checks to boost precision (e.g. grounded semantics as done in (Malmqvist, Yuan, and Nightingale 2024)), extending the approach to other semantics, including admissible, complete, and preferred, where GNNs may be able to exploit local structural patterns even more efficiently. Additionally we aim at lifting the flatness restriction to handle general ABAFs and cover applications where non-flat ABAFs are used (Russo, Rapberger, and Toni 2024). Together, these directions promise more scalable and interpretable reasoning tools for complex argumentative scenarios.

Acknowledgements

Rapberger and Russo were funded by the ERC under the ERC-POC programme (grant number 101189053) while Toni under the EU's Horizon 2020 research and innovation programme (grant number 101020934); Toni also by J.P. Morgan and by the Royal Academy of Engineering under the Research Chairs and Senior Research Fellowships scheme.

References

Biewald, L. 2020. Experiment Tracking with Weights and Biases. Software available from wandb.com.

Bishop, C. M. 2007. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer.

Bondarenko, A.; Dung, P. M.; Kowalski, R. A.; and Toni, F. 1997. An Abstract, Argumentation-Theoretic Approach to Default Reasoning. *Artif. Intell.*, 93: 63–101.

Brody, S.; Alon, U.; and Yahav, E. 2022. How Attentive are Graph Attention Networks? In *Proc. of ICLR*.

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Process. Mag.*, 34(4): 18–42.

Cibier, P.; and Mailly, J. 2024. Graph Convolutional Networks and Graph Attention Networks for Approximating Arguments Acceptability. In *Proc. of COMMA*, 25–36.

Craandijk, D.; and Bex, F. 2020. AGNN: A Deep Learning Architecture for Abstract Argumentation Semantics. In *Proc. of COMMA*, 457–458.

Čyras, K.; Fan, X.; Schulz, C.; and Toni, F. 2018. *Assumption-Based Argumentation: Disputes, Explanations, Preferences. In Handbook of Formal Argumentation.*, chapter 7, 123–145. College Publications.

Čyras, K.; Oliveira, T.; Karamlou, A.; and Toni, F. 2021. Assumption-based argumentation with preferences and goals for patient-centric reasoning with interacting clinical guidelines. *Argument & Computation*, 12(2): 149–189.

De Angelis, E.; Proietti, M.; and Toni, F. 2024. Learning Brave Assumption-Based Argumentation Frameworks via ASP. In *Proc. of ECAI*, 3445–3452.

Dimopoulos, Y.; Nebel, B.; and Toni, F. 2002. On the computational complexity of assumption-based argumentation for default reasoning. *Artif. Intell.*, 141(1): 57–78.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2): 321–357.

Dung, P. M.; Kowalski, R. A.; and Toni, F. 2009. Assumption-Based Argumentation. In Simari, G. R.; and Rahwan, I., eds., *Argumentation in Artificial Intelligence*, 199–218. Springer.

Fan, X. 2018. A Temporal Planning Example with Assumption-Based Argumentation. In *Proc. of PRIMA*, 362–370.

Fandinno, J.; and Lifschitz, V. 2023. Positive Dependency Graphs Revisited. *Theory Pract. Log. Program.*, 23(5): 1128–1137.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. of ICLR*.

Konczak, K.; Linke, T.; and Schaub, T. 2006. Graphs and colorings for answer set programming. *Theory Pract. Log. Program.*, 6(1-2): 61–106.

Kuhlmann, I.; and Thimm, M. 2019. Using Graph Convolutional Networks for Approximate Reasoning with Abstract Argumentation Frameworks: A Feasibility Study. In *Proc. of SUM*, 24–37.

Lagniez, J.; Lonca, E.; and Mailly, J. 2015. CoQuiAAS: A Constraint-Based Quick Abstract Argumentation Solver. In *Proc. of ICTAI*, 928–935.

Lehtonen, T.; Rapberger, A.; Toni, F.; Ulbricht, M.; and Wallner, J. P. 2024. Instantiations and Computational Aspects of Non-Flat Assumption-based Argumentation. In *Proc. of IJCAI*, 3457–3465.

Lehtonen, T.; Rapberger, A.; Ulbricht, M.; and Wallner, J. P. 2021. ACBAR–Atomic-based Argumentation Solver. *ICCMA 2023*, 42(3): 16.

Lehtonen, T.; Rapberger, A.; Ulbricht, M.; and Wallner, J. P. 2023. Argumentation Frameworks Induced by Assumption-based Argumentation: Relating Size and Complexity. In *Proc. of KR*, 440–450.

Li, F.; Wang, H.; and Gupta, G. 2021. grASP: A Graph Based ASP-Solver and Justification System. *CoRR*, abs/2104.01190.

Malmqvist, L.; Yuan, T.; and Nightingale, P. 2024. Approximating problems in abstract argumentation with graph convolutional networks. *Artif. Intell.*, 336: 104209.

Malmqvist, L.; Yuan, T.; Nightingale, P.; and Manandhar, S. 2020. Determining the Acceptability of Abstract Arguments with Graph Convolutional Networks. In *Proc. of COMMA*, 47–56.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. DeepWalk: online learning of social representations. In *Proc. of KDD*, 701–710.

Rapberger, A.; Ulbricht, M.; and Wallner, J. P. 2022. Argumentation Frameworks Induced by Assumption-Based Argumentation: Relating Size and Complexity. In *Proc. of NMR*, 92–103.

Russo, F.; Rapberger, A.; and Toni, F. 2024. Argumentative Causal Discovery. In *Proc. of KR*, 938–949.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1): 61–80.

Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *Proc. of NeurIPS*, 2960–2968.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Proc. of NeurIPS*, 5998–6008.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *Proc. of ICLR*.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous Graph Attention Network. In *Proc. of WWW*, 2022–2032.

Reproducibility Checklist

This paper:

- Includes a conceptual outline and/or pseudocode description of AI methods introduced (**yes**/partial/no/NA)
- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (**yes**/no)
- Provides well marked pedagogical references for less-familiare readers to gain background necessary to replicate the paper (**yes**/no)

Does this paper make theoretical contributions? (**yes**/no)
If yes, please complete the list below.

- All assumptions and restrictions are stated clearly and formally. (**yes**/partial/no)
- All novel claims are stated formally (e.g., in theorem statements). (**yes**/partial/no)
- Proofs of all novel claims are included. (**yes**/partial/no)
- Proof sketches or intuitions are given for complex and/or novel results. (**yes**/partial/no)
- Appropriate citations to theoretical tools used are given. (**yes**/partial/no)
- All theoretical claims are demonstrated empirically to hold. (yes/**partial**/no/NA)
- All experimental code used to eliminate or disprove claims is included. (yes/no/**NA**)

Does this paper rely on one or more datasets? (**yes**/no)
If yes, please complete the list below.

- A motivation is given for why the experiments are conducted on the selected datasets (**yes**/partial/no/NA)
- All novel datasets introduced in this paper are included in a data appendix. (**yes**/partial/no/NA)
- All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (**yes**/partial/no/NA)
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations. (**yes**/no/NA)
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. (**yes**/partial/no/NA)
- All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing. (yes/partial/no/**NA**)

Does this paper include computational experiments? (**yes**/no)
If yes, please complete the list below.

- This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting. (**yes**/partial/no/NA)
- Any code required for pre-processing data is included in the appendix. (**yes**/partial/no).
- All source code required for conducting and analyzing the experiments is included in a code appendix. (**yes**/partial/no)
- All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (**yes**/partial/no)
- All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (**yes**/partial/no)
- If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. (**yes**/partial/no/NA)
- This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. (**yes**/partial/no)
- This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (**yes**/partial/no)
- This paper states the number of algorithm runs used to compute each reported result. (**yes**/no)
- Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. (**yes**/no)
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). (**yes**/partial/no)
- This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. (**yes**/partial/no/NA)

A Correctness of the Extension Reconstruction Algorithm

In this section, we prove the following proposition.

Proposition 5.1. *For an ABAF $\mathcal{D}$ and an assumption a^*, let $\mathcal{D}'$ denote the outcome of* MODIFY$(\mathcal{D}, a^*)$. *If a^* is credulously accepted in $\mathcal{D}$, then the following holds.*

$$\{S \cap \mathcal{A} \mid S \in stb(\mathcal{D}')\} = \{S \setminus \{a^*\} \mid S \in stb(\mathcal{D}), a^* \in S\}$$

To do so, we prove that

1. a correctly guessed accepted assumption can be faithfully removed (correctness of steps 1-3), shown in Proposition A.4, and
2. facts can be faithfully removed (correctness of step 4), shown in Proposition A.2.

We introduce the following two operations. The first one removes facts from an ABAF, the second one removes rejected assumptions (used in Steps 3 and 4).

Definition A.1. *Let $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^-)$ be an ABAF. We define the fact-removal operation $FR(\cdot,\cdot)$ that takes an ABAF $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^-)$ and a fact $r = (p \leftarrow)$ returns $FR(\mathcal{D}, r) = (\mathcal{L}, \mathcal{R}^{rm_1}, \mathcal{A}, {}^-)$ with*

$$\mathcal{R}^{rm_1} = \{head(r') \leftarrow body(r') \setminus \{p\} \mid r' \in \mathcal{R}, r' \neq r\}.$$

Further, we define the assumption-removal operation $AR(\cdot,\cdot)$ that takes an ABAF $\mathcal{D}$ and an assumption b and returns $AR(\mathcal{D}, b) = (\mathcal{L}, \mathcal{R}^{rm_2}, \mathcal{A}^{rm_2}, {}^-)$ with $\mathcal{A}^{rm_2} = \mathcal{A} \setminus \{b\}$ and

$$\mathcal{R}^{rm_2} = \{r \in \mathcal{R} \mid b \notin body(r)\}.$$

We extend these operations to a set of rules/assumptions by iteratively applying the operations to each element in the set.

We start by proving 2. and show that facts can be faithfully removed (step 4).

Proposition A.2. *Let $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^-)$ be an ABAF and let $r = (p \leftarrow) \in \mathcal{R}$. Let $C = \{a \in \mathcal{A} \mid \overline{a} = p\}$. Then $stb(\mathcal{D}) = stb(FR(AR(\mathcal{D}, C), r))$.*

Proof. Let $\mathcal{D}' = FR(AR(\mathcal{D}, C), r)$ with assumption set $\mathcal{A}'$ and rule set $\mathcal{R}'$.

First note that there is a 1-1 correspondence between arguments constructed from S in $\mathcal{D}$ and D', for each set S of assumptions which does not contain elements from C. That is, for each $S \subseteq \mathcal{A}$ with $S \cap C = \emptyset$, $S \subseteq q$ is an argument in $\mathcal{D}$ iff $S \vdash q$ is an argument in $\mathcal{D}'$:

($\Rightarrow$) Let $S \vdash q$ be an argument in $\mathcal{D}$. Let $R \subseteq \mathcal{R}$ such that $S \vdash_R q$ in $\mathcal{D}$. By assumption, $b \notin body(r')$ for each $b \in C$ and for each $r' \in R$, thus, r' is not deleted in $\mathcal{R}'$. In case that $r \in R$, then r is 'cut out' from the tree derivation in the rule modification: each occurrence from p is removed from the body of each rule, and r is removed. Let R' denote the modified rules. We can construct an argument $S' \vdash_{R' \setminus \{r\}} \overline{c}$.

($\Leftarrow$) For the other direction, $S \vdash q$ denote an argument in $\mathcal{D}'$ and let $R \subseteq \mathcal{R}'$ denote a rule set such that $S \vdash_R q$ in $\mathcal{D}'$. Let R° denote the set of rules corresponding to R in $\mathcal{D}$. In case $p \notin body(r')$ for each rule $r \in R^\circ$ then $R = R^\circ$ and $S \vdash_R q$ is a tree-derivation in $\mathcal{D}$. Otherwise, $S \vdash_{R^\circ \cup \{r\}} q$ is a tree-derivation in $\mathcal{D}$. In both cases, we obtain that $S \vdash q$ is an argument in $\mathcal{D}$.

- First, let $S \in stb(\mathcal{D})$. Then $b \notin S$ for each $b \in C$ since each b is attacked by a fact. Thus, as shown above, we can construct the same arguments from S in $\mathcal{D}$ and D'. Consequently, S is conflict-free in $\mathcal{D}'$ (otherwise, S would attack itself in $\mathcal{D}$ as well) and attacks all assumptions in $\mathcal{A}' \setminus S$ (since $\mathcal{A}' \subseteq \mathcal{A}$). We obtain $S \in stb(\mathcal{D}')$.
- Now, let $S \in stb(\mathcal{D}')$. S does not contain arguments from C. Then S is conflict-free in $\mathcal{D}$ since there is a 1-1 correspondence between the arguments constructed from S in $\mathcal{D}$ and $\mathcal{D}'$ (thus, if there is a conflict in S in $\mathcal{D}$ then S attacks itself in $\mathcal{D}$' as well). Moreover, S attacks each assumption in $\mathcal{A}' \setminus S$ in D by the same reasoning and since S is stable in $\mathcal{D}'$. Each remaining assumption is attacked by the fact $r = (p \leftarrow)$. Thus, S is stable in $\mathcal{D}$. □

This shows that a fact can be removed without altering the stable sets. Iterative application of Proposition A.2 shows correctness of step 4.

Next we show 2. and prove that that steps 1-3 are correct. We define the ABAF modification *accept assumption* $AA(\mathcal{D}, a)$ below.

Definition A.3. *Let $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^-)$ be an ABAF, let $a \in \mathcal{A}$, let $C = \{c \in \mathcal{A} \mid \overline{c} = a\}$ and let $R_{\overline{a}} = \{r \in \mathcal{R} \mid head(r) = \overline{a}\}$. We define $AA(\mathcal{D}, a) = (\mathcal{L}, \mathcal{R}', \mathcal{A}', {}^-)$ with*

$$\mathcal{A}' = (\mathcal{A} \cup \{d_r \mid r \in R_{\overline{a}}\}) \setminus (C \cup \{a\})$$

and

$$R' = \{head(r) \leftarrow body(r) \setminus \{a\} \mid r \in \mathcal{R} \setminus R_{\overline{a}}, C \cap body(r) = \emptyset\} \cup \{\overline{d_r} \leftarrow (body(r) \cup d_r) \setminus \{a\} \mid r \in R_{\overline{a}}, C \cap body(r) = \emptyset\}.$$

We show that this ABAF satisfies the equation in Proposition 5.1.

Proposition A.4. *Let $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^-)$ be an ABAF, let $a \in \mathcal{A}$ be credulously accepted with respect to stable semantics in $\mathcal{D}$, and let $\mathcal{D}' = AA(\mathcal{D}, a)$. Then the following is satisfied.*

$$\{S \cap \mathcal{A} \mid S \in stb(\mathcal{D}')\} = \{S \setminus \{a\} \mid S \in stb(\mathcal{D}),\ a \in S\}.$$

Proof. Let $C = \{c \in \mathcal{A} \mid \overline{c} = a\}$ denote all assumptions with a as its contrary. Let $\mathcal{D}' = RAA(\mathcal{D}, a) = (\mathcal{L}, \mathcal{R}', \mathcal{A}', {}^-)$.

By assumption, the ABAF $\mathcal{D}$ has a stable extension (since a is credulously accepted). Moreover, by construction, each assumption of the form d_r can be accepted in case $\mathcal{D}'$ has a stable extension. Thus, each stable extension contains all $\{d_r \mid r \in R_{\overline{a}}\}$.

- Let $S \in stb(\mathcal{D}')$. We show that $S' = (S \cap \mathcal{A}) \cup \{a\} \in stb(\mathcal{D})$.
  S' is conflict-free in $\mathcal{D}$: towards a contradiction, suppose S' is not conflict-free. Let $T \vdash_R \overline{b}$ for some $T \subseteq S'$, $R \subseteq \mathcal{R}$, $b \in S'$, in $\mathcal{D}$. Note that $body(r) \cap C = \emptyset$ for each $r \in R$ since $C \cap S' = \emptyset$. We proceed by case distinction. Case 1: $head(r) \neq \overline{a}$ for all $r \in R$. Let R' denote the modified rules in $\mathcal{D}'$ corresponding to R. Each rule $r \in R$ either stays unchanged (in case $a \notin body(r)$) or the assumption a is removed from the body. We can construct

Table 1: Parameter ranges for ICCMA benchmarks and additional synthetic data

Parameter	ICCMA	Generated
Acceptance Rate (with at least one ext)	36.7% (60.3%)	32.5% (53.4%)
Number of atoms	25, 100, 500, 2000, 5000	25, 50, 75, 100, 250, 500, 750, 1000, 2000, 3000, 4000, 5000
Assumption proportion	10%, 30%	25%, 40%, 50%, 60%, 75%
Max. rules per derived atom	5, 10	2, 4, 8, 16
Max. rule-body length	5, 10	2, 4, 8, 16
Cycle cap	—	0.01, 0.03, 0.05, 0.07, 0.09, 1.0
Total instances (post-filtering)	400 (380)	48 000 (19 500)

an argument in $\mathcal{D}'$ based on $T \setminus \{a\}$ using R' that derives $\overline{b}$. Thus, b is attacked by S in $\mathcal{D}'$, contradiction to S being stable in D'.

Case 2: there is some $r \in R$ with $head(r) = \overline{a}$. Wlog, we can assume that the top-rule r of the argument satisfies $head(r) = \overline{a}$ (otherwise choose an appropriate sub-argument). By definition, this induces a rule $\overline{d_r} \leftarrow (body(r) \cup d_r) \setminus \{a\}$ in $\mathcal{D}'$. Every element in $body(r)$ can be derived from S in $\mathcal{D}'$ (this can be proven analogous to Case 1). Thus, we can construct an argument from assumptions in S (recall that $\{d_r \mid r \in R_{\overline{a}}\} \subseteq S$) that derives $\overline{d_r}$. Thus, S is conflicting in $\mathcal{D}'$, contradiction to the assumption $S \in stb(\mathcal{D}')$.

We have shown that S' is conflict-free in $\mathcal{D}$. It remains to show that S' attacks each assumption in $\mathcal{A} \setminus S'$ in $\mathcal{D}$. Let $b \in \mathcal{A} \setminus S'$. The assumption b is not contained in S in $\mathcal{D}'$.

Case 1: $b \notin \mathcal{A}'$. Then $b \in C$. Since $a \in S'$, b is attacked.

Case 2: $b \in \mathcal{A}'$. Then b is attacked by S in $\mathcal{D}'$. That is, there is a tree-derivation $T \vdash_R \overline{b}$ in D' with $T \subseteq S$. We observe that (i) $\overline{a}$ does not appear in the tree derivation (otherwise, we can construct an argument with conclusion $\overline{a}$ from assumptions in S in $\mathcal{D}'$, contradiction to S being stable in $\mathcal{D}'$); and (ii) assumptions of the form d_r do not appear in the tree derivation (otherwise we can derive a contradiction to conflict-freeness of S like in Case 2). For each $r \in R$, either $r \in \mathcal{R}$ or $(head(r) \leftarrow body(r) \cup \{a\}) \in \mathcal{R}$. We thus can use the rules in R to construct a tree-derivation from $T \cup \{a\}$ with conclusion $\overline{b}$ in $\mathcal{D}$ and obtain $S' \in stb(\mathcal{D})$, as desired. .

- Let $S \in stb(\mathcal{D})$ with $a \in S$. Let

$$S' = (S \setminus \{a\}) \cup \{d_r \mid r \in R_{\overline{a}}\}.$$

We show that $S' \in stb(\mathcal{D}')$.

S' is conflict-free: towards a contradiction, suppose there is a tree derivation $T \vdash_R \overline{b}$ for some $b \in S'$ in $\mathcal{D}'$.

Case 1: b is of the form d_r for some $r \in R_{\overline{a}}$. Wlog let d_r be the only occurrence of assumptions of this form (otherwise, choose an appropriate sub-argument). Then each $p \in body(r)$ can be derived from assumptions in S. Thus, we can construct an argument with conclusion $\overline{a}$ from S, contradiction to conflict-freeness of S in $\mathcal{D}$.

Case 2: $b \in \overline{\mathcal{A}}$. The only possible rule modifications to obtain the rules in R (from rules in $\mathcal{R}$) is the removal of assumption a. That is, for each rule $r \in R$, either $r \in \mathcal{R}$ or $(head(r) \leftarrow body(r) \cup \{a\}) \in \mathcal{R}$. Let R' denote the rules corresponding to R in $\mathcal{D}$. We can construct an argument in $\mathcal{D}$ based on $T \cup \{a\}$ (in case $R \neq R'$) or T (in case $R = R'$) that derives $\overline{b}$. Thus, b is attacked by S in $\mathcal{D}$, contradiction to conflict-freeness of S in $\mathcal{D}$.

We have shown that S' is conflict-free in $\mathcal{D}'$. It remains to show that S' attacks all remaining assumptions in $\mathcal{D}'$. By definition of S', all remaining assumptions are contained in $\mathcal{A}$ as well. We thus can proceed analogous to Case 2 to construct arguments with the appropriate conclusions and conclude that each remaining assumption is attacked by S'. Thus, S' is stable in D'. □

By Proposition A.4, and by iterative application of Proposition A.2, we obtain that the equation in Proposition 5.1 is correct. This implies that the algorithm presented in Section 5 successfully returns a stable extension, provided the predictions are correct.

B Details on Experiments

Here we provide additional details for the experiments presented in Section 6. We cover data, hyperparameter tuning and additional details about performance, including statistical tests for the observed differences amongst methods.

B.1 Data

We began with the 400 ABAFs from the ICCMA 2023 benchmark,[6] each parameterised by the number of atoms, the proportion of those atoms that are assumptions, the maximum number of rules per derived atom, and the maximum rule-body length. After running ASPForABA with a 10 min timeout per instance, we retained 380 ABAFs annotated with credulous accepted labels for each assumption.

To augment this corpus, we used the ABAF generator in (Lehtonen et al. 2024) to produce 48,000 additional "flat" frameworks, extending each of the four original parameters as shown in Table 1. We then filtered out any instances that timed out during labelling (yielding 47,794), and from these we sampled a curated subset of 19,500 ABAFs so as to closely match the acceptance rates observed in the ICCMA data (32.5 % overall, 53.4 % among solvable instances). In particular, we balanced the generated set to include 13,000

[6] Available at https://zenodo.org/records/8348039

Table 2: Tested hyperparameter ranges and best configurations for GCN and GAT

Parameter	Values tested	GCN (best)	GAT (best)
Validation F1 Score	—	0.6682	0.6709
Embedding dimension	16, 32, 64, 128, 256	32	64
Hidden dimension	32, 64, 128, 256, 512	32	64
Number of layers blocks	2–10	10	10
Dropout rate	0–0.5 (uniform)	0.0294	0.2198
Learning rate	10^{-2}–10^{-4} (uniform)	0.0086	0.0066
Batch size	16, 32, 64, 128	128	64
Class imbalance weight	1.25, 1.5, 1.75, 2.0, 2.25	2.25	1.75
Classification threshold	0.20–0.85 (step 0.05)	0.50	0.45
Early stopping patience	10, 30, 50, 70, 90	10	30

ABAFs with at least one accepted assumption and 6,500 with none. Finally, to support fair evaluation of both our GNN models and the AFGCNv2-based baseline—which cannot predict acceptance for ABAFs with more than 100 atoms—we stratified the entire collection into three groups (small ICCMA: 25–100 atoms; full ICCMA; and Generated) and performed a 75/25 train–test split within each group. Our augmented dataset is provided in our repository.[7]

Computing infrastructure All the results were ran on NVIDIA(R) GeForce RTX 4090 GPU with 24GB dedicated RAM.

B.2 Evaluation Metrics

We evaluated the classification results using four metrics:

- Precision = TP/(TP + FP)
- Recall = TP/(TP + FN)
- F1 = 2×(Precision×Recall)/(Precision+Recall)
- Accuracy = TP/(TP + FP + TN + FN)

Precision and Recall measure the proportion of correct classifications based on the predicted classes and the true ones, respectively. In the formulae, True Positive (TP) is the number of estimated assumptions correctly credulously accepted; False Positive (FP) is an assumption which is credulously accepted but is not in any extension. True Negative (TN) and False Negative (FN) are assumptions that are not in any extension and are correctly (resp, incorrectly) classified as not accepted. F1 score is the harmonic mean of precision and recall. Finally, accuracy just looks at the TP rate over the full assumption set. For the credulously acceptance classification the metrics are calculated over the full set of assumption, while for the extension-reconstruction task they assumption set is restricted to the assumptions in any of the valid extensions.

B.3 Hyperparameter Tuning

We conducted hyperparameter optimisation separately for the ABAGCN and ABAGAT models using Bayesian search over the ranges reported in Table 2. For each candidate configuration we trained on the 90 % of the 75 % training split and selected the model checkpoint with the highest F1 on the 10 % held-out validation set, employing early stopping with patience drawn from $\{10, 30, 50, 70, 90\}$. Our search revealed that the attention-based variant (ABAGAT) attained a marginally higher peak F1 (0.6709 vs. 0.6682) by leveraging a larger embedding and hidden dimension (64 vs. 32), higher dropout ($\simeq$0.22 vs. 0.03), and a smaller batch size (64 vs. 128). Interestingly, both models benefited from deeper architectures—ten message-passing layers—as well as non-trivial class-imbalance weighting (1.75-2.25) and mid-range learning rates ($6 - 9e^{-3}$). We also observed that the attention mechanism appeared to regularize the model, allowing for higher dropout and greater early-stopping patience without degrading validation performance.

B.4 Extensions Accuracy

Beyond node-level classification, we evaluated how well our extension-reconstruction algorithm (Section 5) recovers full stable extensions. We plot the results discussed in the main text is Figure 3 where we show the F1 of reconstructed extensions as a function of ABAF size. On very small frameworks (10 atoms), our method achieves high fidelity (F1$\sim$0.85), but performance degrades steadily as the number of atoms grows, falling to $\sim$0.60 for 50–100 atoms and plateauing around 0.58 for larger instances. This trend reflects the compounding effect of early misclassifications: a single mistaken assumption in the initial GNN output can

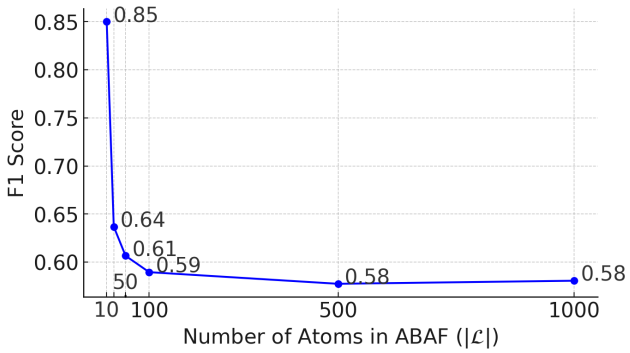


Figure 3: F1 score of the extension by ABAF size.

[7] Available at https://anonymous.4open.science/r/GNN4ABA

Scenario	Model	F1 Score	Accuracy	Precision	Recall
Small ICCMA	AFGCNv2	0.504±0.020	0.435±0.020	0.377±0.018	0.758±0.022
	ABAGCN	**0.721±0.103*****	**0.730±0.071*****	**0.625±0.133*****	**0.866±0.053*****
	ABAGAT	**0.735±0.089*****	**0.746±0.055*****	**0.639±0.120*****	**0.877±0.049*****
All ICCMA	ABAGCN	0.687±0.119	0.710±0.093	0.569±0.145	0.899±0.037
	ABAGAT	0.713±0.116	0.728±0.094	0.580±0.141	**0.957±0.024****
ICCMA+Gen	ABAGCN	0.656±0.009	0.728±0.005	0.548±0.012	**0.816±0.004*****
	ABAGAT	0.658±0.009	**0.735±0.004****	**0.558±0.012 .**	0.803±0.005

(a) Mean and Standard Deviation ($\mu \pm \sigma$)

Scenario	Metric	Method 1	Method 2	t	p-value
Small	F1	ABAGAT (0.735 ± 0.089)	ABAGCN (0.721 ± 0.103)	0.329	0.746
Small	F1	ABAGAT (0.735 ± 0.089)	AFGCNv2 (0.504 ± 0.020)	8.029	0.000***
Small	F1	ABAGCN (0.721 ± 0.103)	AFGCNv2 (0.504 ± 0.020)	6.579	0.000***
Small	Accuracy	ABAGAT (0.746 ± 0.055)	ABAGCN (0.730 ± 0.071)	0.557	0.585
Small	Accuracy	ABAGAT (0.746 ± 0.055)	AFGCNv2 (0.435 ± 0.020)	16.669	0.000***
Small	Accuracy	ABAGCN (0.730 ± 0.071)	AFGCNv2 (0.435 ± 0.020)	12.728	0.000***
Small	Precision	ABAGAT (0.639 ± 0.120)	ABAGCN (0.625 ± 0.133)	0.253	0.803
Small	Precision	ABAGAT (0.639 ± 0.120)	AFGCNv2 (0.377 ± 0.018)	6.839	0.000***
Small	Precision	ABAGCN (0.625 ± 0.133)	AFGCNv2 (0.377 ± 0.018)	5.849	0.000***
Small	Recall	ABAGAT (0.877 ± 0.049)	ABAGCN (0.866 ± 0.053)	0.482	0.636
Small	Recall	ABAGAT (0.877 ± 0.049)	AFGCNv2 (0.758 ± 0.022)	7.020	0.000***
Small	Recall	ABAGCN (0.866 ± 0.053)	AFGCNv2 (0.758 ± 0.022)	5.989	0.000***
Full	F1	ABAGAT (0.713 ± 0.116)	ABAGCN (0.687 ± 0.119)	0.493	0.628
Full	Accuracy	ABAGAT (0.728 ± 0.094)	ABAGCN (0.710 ± 0.093)	0.432	0.671
Full	Precision	ABAGAT (0.580 ± 0.141)	ABAGCN (0.569 ± 0.145)	0.183	0.857
Full	Recall	ABAGAT (0.957 ± 0.024)	ABAGCN (0.899 ± 0.037)	4.183	0.001**
Full+Gen	F1	ABAGAT (0.658 ± 0.009)	ABAGCN (0.656 ± 0.009)	0.703	0.491
Full+Gen	Accuracy	ABAGAT (0.735 ± 0.004)	ABAGCN (0.728 ± 0.005)	3.822	0.001**
Full+Gen	Precision	ABAGAT (0.558 ± 0.012)	ABAGCN (0.548 ± 0.012)	1.829	0.084 .
Full+Gen	Recall	ABAGAT (0.803 ± 0.005)	ABAGCN (0.816 ± 0.004)	-6.500	0.000***

(b) Welch t-test results for each metric and scenario.

Table 3: (a) Mean and Standard Deviation ($\mu \pm \sigma$) summary and (b) full t-test breakdown for ICCMA small (25/100 element files), full (all element files), and ICCMA+Gen datasets. Sample sizes: 30/40 (small), 71/95 (full), 3727/4970 (ICCMA+Gen); Bootstrap samples: 10. Significance codes: 0‘***’0.001‘**’0.01‘*’0.05‘.’0.1.

cascade through the constructive search steps and disproportionately harm both precision and recall. We envisage good opportunities to address this via lookback and lookahead error correction strategies to prevent error propagation.

B.5 Statistical Tests

To quantify the reliability of the performance differences among our models, we conducted Welch's two-sample t-tests on each metric in each evaluation scenario.

Table 3a reproduces the mean (μ) and standard deviation (σ) for F1, accuracy, precision, and recall in each scenario (Small ICCMA, All ICCMA, and ICCMA+Generated), exactly as plotted in Figure 2 of the main text. In this table each "$\mu \pm \sigma$" entry carries a superscript significance code—$***$ for $p < 0.001$, $**$ for $p < 0.01$, $*$ for $p < 0.05$, and **.** for $p < 0.1$—and is typeset in **bold** only when the top-ranked model's mean is significantly higher than the runner-up at one of these levels. Table 3b provides the full pairwise comparison details that underpin the annotations in Table 3a. For each scenario and metric it reports the two methods being compared, the Welch t-statistic, the exact p-value, and the same significance codes. This detailed breakdown allows readers to verify the strength and direction of every statistical difference that has been collapsed into the boldface annotations of the first table.