Simulating the Structure of Collaborative Discourse

Blinded for Review

October 9, 2025

Abstract

Researching computer-supported collaborative learning typically requires observing collaboration and recording those observations as data. The process of observation and recording is difficult and may leads result that may not generalise well. We describe and validate a method generating synthetic data that mimics the structure of collaborative discourse. Our results show that the method can reproduce the structure of real collaborative discourse data. We demonstrate the utility of this method by using it as part of a sensitivity analysis to compare the impact of highly ordered collaborative discourse on the results of two common analytic techniques. The results show that even small amounts of explicit ordering can lead to significant differences between model outputs.

1 Introduction

Collaboration is vital to society. The problems we face as persons, groups, organisations, and cultures are often too complex and ill-defined to solve alone. At the same time, we are born into social contexts that force collaboration and, to a large extent, define how we learn and develop. Understanding what makes effective and ineffective collaboration is thus key to human progress.

Our efforts to systematically understand collaboration have taken many forms. Here, we focus on two kinds: investigations of collaborative problem solving (CPS) and computer supported collaborative learning (CSCL). CPS research looks at how social and cognitive factors combine to influence solutions to shared problems (Hesse et al., 2015). CSCL research examines how collaboration, mediated by technology, relates to human learning (Baker and Reimann, 2025). Both place an emphasis on examining interactions among humans, tools, and their broader environment.

Analysis of these interactions typically involves observations of some joint activity that produce records of discourse—things that groups say or do that serve a communicative function. While these observations are critical, as any CPS or CSCL research knows, they are nontrivial to obtain. Studies must be designed, ethics approved, participants recruited, and data collected. Each of these steps is labour intensive. Moreover, the the participants we have and the data they produce necessarily constrain the conclusions we can draw. If we are unable to collect data from certain demographics, if our studies are underpowered, or if data collection is disrupted, we are stuck. In situations such as these, the typical recourse for collaboration researchers is to either live with the limitations or to retry the data collection process.

While we should still place a premium on real observations of collaborative discourse, this paper offers a complementary solution, namely, simulation. With appropriate techniques, collaborative discourse can simulated under desired conditions and manipulated to test the sensitivity of the data to different conditions. Despite wider use in the social sciences (Gilber and Troitzsch, 2005), such simulations are rarely used in CPS or CSCL studies. In this paper, we introduce a novel computation simulation methodology. We describe the theoretical constructs that inform the design of the method, validate the method by comparing its outputs to real data collected from a CSCL environment, and demonstrate how the method can be used to test the sensitivity of widely used analytic techniques.

2 Background

2.1 Analysing Collaboration

Collaboration is a relatively well-studied phenomenon. Methods for studying it typically involve observing joint activity that has been explicitly structured for research purposes or exists in the everyday world. As part of these observations, records of collaborative discourse are produced. These records often take the form of transcripts of group talk, but increasingly are coming to include a variety of multimodal data including gestures, tool manipulations, gazes, spatial positioning, and so on (Martinez-Maldonado et al., 2021; Shaffer et al., 2025).

Analysis usually proceeds by segmenting the data and adding an interpretive layer, such as *codes* (Shaffer and Ruis, 2021). After the coding has been validated and applied to the data, in quantitative and mixed methods approaches, much of the subsequent work is done with the codes. Traditional analytics approaches often involve some sort of coding-and-counting procedure (Chi, 1997). For example, Zhou and colleagues (2025) observed 5th-grade students' peer feedback conversations in a CSCL environment. They coded these conversations for episodes of disagreement and whether those disagreements were direct or ”soft”—i.e., implied. Using a frequency analysis of the codes, they found that the majority of disagreements in their data were soft.

Significant emphasis has also been placed on understanding sequences or patterns of codes in the data. Techniques like process modelling (Reimann et al., 2009), social network analysis (Moser et al., 2013), and variants of epistemic network analysis (Shaffer et al., 2016; Tan et al., 2023) are now widely applied techniques in the CPS and CSCL literature. For example, Zabolotana and colleagues (2025) investigated how different task designs impacted collaborative knowledge construction and group-level regulation. To do so they observed and recorded students' collaborative utterances during one physics task and one robotics task. They coded the data for five phases of collaborative knowledge construction: share/compare, disagree/dissonance, negotiate/co-construct, test/evaluate/modify, and agree/apply, as well as two kinds of regulatory behaviours, emotional/motivational and cognitive. Using ordered network analysis (Tan et al., 2023), they compared the patterns of ordered connections between codes present in the data between the two tasks.

While sophisticated techniques techniques exist for making the process of recording, segmenting, coding, and analysing collaborative discourse more efficient, the processes of eliciting collaborative discourse remains labour intensive. Studies need to be designed, ethics applications need to be approved, and participants need to be recruited. Even then, as anyone who has observed collaborative activity before knows, participants may act in unexpected ways—they may subvert the study design or fail to engage in collaborative behaviour. And after the observations have been recorded, coded, and analysed, any findings are limited by the qualities of the data collected.

For example, in their study on disagreement, Zhou and colleagues (2025) noted that they did not explore the potential impact of other factors known to play a role in how disagreements are expressed and resolved, such as discussion topics, prior knowledge, and gender dynamics. Similarly, Zabolotana and colleagues (2025), noted that one of the theorised stages of cognitive knowledge construction, agree/apply, was excluded from their analysis due to low frequency in their data. Notice that both of these issues stem from the specific data collected: Zhou and colleagues could not say anything about different discussion topics because they only collected data on one kind of topic; Zabolanta and colleagues could not explore the role that the final stage of collaborative knowledge construction played because they did not observe it enough. While these issues do not invalidate their findings, they limit them[1]. Of course, all findings are limited by the data at hand and the analytic techniques used. Researchers either choose to live with these limitations or plan to address them in future work, which often requires collecting more data. Another solution that is less applied in CPS and CSCL studies is simulation—generating synthetic data that allows us to test hypotheses dependent on unseen data.

2.2 Simulation

The term simulation can carry several meanings in the context of CPS and CSCL research. Here, we distinguish between *experiential* simulations—structured activities in which participants play some role in a digital or physical environment that is meant to mimic the salient parts of some real context—and *computational* simulations—mathematical formulae or algorithms that generate synthetic data under pre-defined conditions. Thus, pedagogical scenarios like virtual internships (Chesler et al., 2015), where learners play the role of interns at a fictitious

[1]Note that we choose these two studies not to critique them, but because they are recent exemplars of CSCL/CPS studies.

companies to learn about design thinking is a kind fo experiential simulation. In contrast, systems like *NetLogo* (Tisue and Wilensky, 2004) are designed to create and run computational simulations that produce data under specific conditions.

While there are several kinds of computational simulations, our focus is on *agent-based modelling* (ABM)—models in which autonomous entities ("agents") interact with one another and their environment according to predefined rules (Gilber and Troitzsch, 2005). Each agent embodies individual characteristics, decision heuristics, or traits, and through repeated interactions, complex group-level patterns can emerge. ABM is particularly useful for studying systems where aggregate outcomes depend on heterogeneous individuals and dynamic interactions, allowing researchers to explore sensitivity to initial conditions, test counterfactual scenarios, and generate synthetic data when large-scale empirical observations are difficult to obtain. ABMs have been used in social science research to understand the emergent properties of groups. For example, Jager and colleagues (Jager et al., 2001), used ABM to study the likelihood of conflicts arising between different groups. More recently, Zoller and colleagues (2021) used ABM to study the impact of different social network structures on the role behaviours of collaborative software developers.

Despite their utility, few computational simulation methods—agent-based or otherwise—have been applied to studies of CSCL or CPS in the educational literature. Ironically, one of the first applications of computational simulation was conducted by Hutchins (1995) in his seminal work *Cognition in the Wild.* As an example of the power of computational methods, Hutchins modelled agents as constraint-satisfaction networks and built communities of these networks to study how different communication patterns impact confirmation bias.

Bergner and colleagues (2016) used ABM to investigate the dynamics of CPS and assess whether simulations could generate data useful for subsequent analyses. Using the *Subarctic Survival Situation* as a task template, they modelled small groups of agents who communicated to iteratively rank lists of items according to survival importance. While Bergner and colleagues simulation demonstrated that ABM can both reproduce realistic CPS processes and yield synthetic data for refining assessments of collaborative skills, the kind of data generated by the method corresponds to a very narrow range of discourse moves. Specifically, agents could make only two possible actions that showed up in the data—showing their list and changing their list. Of course, real collaborative discourse is much more complex. And even when simplifying via a coding process, the range of actions that CSCL researchers explore maybe to be higher. Zabolotana and colleagues' (2025) study mentioned above, for instance, used seven codes to describe the discourse they observed.

More recently, Swiecki and colleagues (2022) proposed and tested an ABM capable of simulating a larger variety of simulated discourse moves. Rather than simulating the content of the discourse, their method operates at the level of codes, using data-determined or user specified social and epistemic networks to drive the simulation process. Their results showed that their method was able to replicate the code patterns presented in real discourse data.

While a step forward, Swiecki and colleagues' simulation had two important limitations. First, the method assumes a model of discourse based on the influence of *common ground* (Clark, 1996) operationalised as the prior discourse moves made by the group. In typical analyses of collaborative discourse, researchers choose some window size of prior moves that is the most influential (Ruis et al., 2019). While empirical studies have not examined the range of window sizes used in the literature, our own studies have used window sizes as small as one prior move and as large as seven or more. Despite the potential influence of larger representations of common ground, Swiecki and colleagues simulation procedure could only accommodate a window size of one prior discourse move, similar to a first-order Markov modelling approach.

Second, collaborative discourse tends to be coded in one of two ways Graesser et al. (1992). Monothetic coding schemes may assign one and only one code to a discourse move or segment of data. Polythetic coding schemes may assign multiple codes for the same segment. While prior work has not investigated the relative prevalence of both types, our own experience suggests that polythetic coding schemes are not uncommon. Despite this, Swiecki and colleagues' simulation procedure could only produce monothetically coded discourse.

To address these limitations, we extend Swiecki and colleagues' simulation methodology to account for extended representations of common ground—i.e., window sizes greater than one discourse move—polythetic coding schemes. Specifically, we ask the following research questions:

1. Can the updated simulation method accurately replicate coded discourse from a real CSCL scenario?
   (a) Can the updated simulation method produce monothetically coded discourse with extended representations of common ground?
   (b) Can the updated simulation method produce polythetically coded discourse with extended representations of common ground?

2.3 Sensitivity Analysis

Replicating salient features of real collaborative discourse is important for warranting the validity of the simulation method, but it does not suffice to show its utility. The prior works by Hutchins, Bergner and colleagues, and Swiecki and colleagues demonstrated the power of computational simulations by using them to explore a variety of collaborative conditions.

Hutchins modelled individuals as constraint satisfaction networks, where each network was defined by individual parameters such as initial connection patterns, node values, or activations, and environmental inputs. Hutchins began his simulation with many identical networks, differing slightly in initial activations and all receiving similar environmental inputs. He then varied *persuasiveness*, operationalised as the the strength of the inter-network connections—while keeping other things equal. At zero persuasiveness each network settles to stable activation configurations, or *interpretations*, that are closest to their original predispositions. Low levels of persuasiveness makes individuals move quickly toward their initially preferred interpretations. Moderate levels tend to pull borderline individuals—those initially between alternative interpretations—toward others' interpretations. At very high persuasiveness levels the whole community rushes to a single shared interpretation and becomes extremely resistant to new evidence. Overall, he concluded that more communication is not always better. Very high levels may lead to rigid decision making and confirmation bias.

In Bergner and colleagues study, agent traits were drawn from distributions representing three levels of talkativeness (likelihood of showing one's ranking), agreeableness (willingness to adopt alternative rankings), and critical thinking (accuracy in evaluating competing rankings). A global knowledge-sharing parameter, also varied at three levels, determined how exhaustively agents compared list pairs. This design produced more than 40,000 simulation runs, with outcomes tracked using correlation to the ground-truth list and team-level measures of consensus, gain (improvement over the average individual), and synergy (improvement over the best individual). Results indicated that higher talkativeness and agreeableness consistently increased gain scores and consensus rates, although talkativeness lengthened the duration of interactions while agreeableness shortened them. Critical thinking contributed modestly to gains and was the key driver of synergy, but it also lowered consensus rates by discouraging premature agreement. Knowledge sharing raised both gain and synergy scores but had little effect on consensus or task duration.

Similarly, Swiecki and colleagues used their simulation procedure to explore the impact of different levels of *dissimilarity* of shared information and *interactivity* among teammates on conversation patterns pre-and post a jigsaw (Aronson, 1978) intervention. Critically, the simulation allowed them to examine a greater array of dissimilarity and interactivity values than those present in a real dataset. Analysing the simulated data, they found that as dissimilarity and interactivity increases, patterns of connections between discourse codes align more with what we would expect from a post-jigsaw team—that is, connections among a variety of topics unavailable to them before the jigsaw. These results suggest that increasing the dissimilarity of information shared and the interactivity among teammates—perhaps through collaboration scripts—can increase the effect of the jigsaw pedagogy.

One way to collectively describe these three studies is through the lens of *sensitivity analysis*—a systematic approach for examining how variations in model inputs influence model outputs (Christopher Frey and Patil, 2002). It serves to identify which assumptions or parameters most strongly affect results, to evaluate the robustness of conclusions under uncertainty, and to guide data collection by highlighting inputs that require more precise estimation.

As the prior examples show, computational simulation methods are integral to sensitivity analysis, particularly in complex systems where analytical solutions are intractable. By repeatedly sampling from specified parameter ranges or probability distributions, simulations generate ensembles of model outcomes that reflect uncertainty in the inputs. Comparing how outputs vary across these ensembles allows researchers to quantify the relative importance of parameters and assess the stability of key findings.

To demonstrate the utility of the improved version of Swiecki and colleagues simulation, we conduct a sensitivity analysis to understand how different communication patterns relate to important modelling choices in CSCL and CPS research.

Several authors have argued that accounting for the temporal nature of collaborative discourse is critical to valid models of CPS (Kapur, 2011; Reimann et al., 2009; Halpin and von Davier, 2017). Operationally, this typically means that when researchers analyse collaborative discourse, they account for the sequence in which discourse moves occur using methods like lag sequential analysis, pattern mining, or ordered network analysis. However, just because something occurs in a particular order, does not mean that the order is important to model. Modelling is about re-representing the complexity of world in terms of some simpler representation. Only the features salient to the researcher's line of enquiry should make it into the simplified representation. Prior work has shown that the order of particular discourse moves may not always be relevant in collaborative scenarios (Swiecki et al., 2019). Nonetheless, there may be scenarios in which order is very relevant.

It is up to the researcher to make and justify these modelling decisions. And like all decisions, they have consequences. If order is important and it is left out, the validity of the results and subsequent interpretations is questionable. If order is not important and it is left in the model, the model may be overly complex, difficulty to interpret, and difficult to convey it's meaning to other stakeholders.

To illustrate one use case for our simulation procedure, we use it to vary the degree of ordering—that is, the likelihood preferred orders of code-occurrences– present in collaborative discourse data. We then compare the results of two widely used analytic techniques, ENA and ONA, on these data to determine the degree of ordering that leads to statistically significant different between the model outputs. This sensitivity analysis aims to provide CSCL and CPS researchers additional evidence and guidance for when it may be necessary to explicitly account for the order of coded discourse moves in their models thus adopt a more complex modelling technique. Specifically, we ask the following research question:

2. What degree of ordering of discourse moves is important to account for in models of CSCL and CPS?

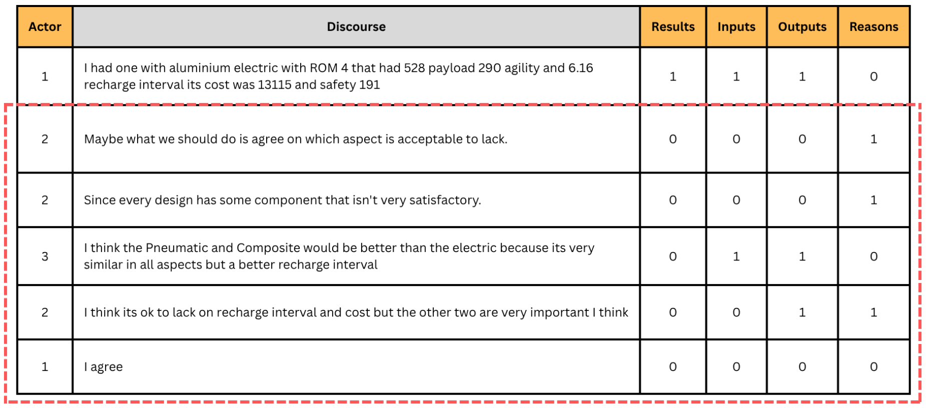
3 Methodology

3.1 Theoretical Basis

Our simulation procedure is strongly influenced by Dillenbourg's (1999) characterisation of collaborative learning in terms of situations, interactions, processes, and effects. According to Dillenbourg, the degree to which situations are collaborative is determined by the symmetry of action, knowledge, and status between groupmates. Collaborative learning situations can also be characterised by the extent to which the interaction between groupmates influences their cognitive processes. Thus, interactions in collaborative learning situations trigger certain learning mechanisms or processes. Finally, the effects of collaboration are generally things that researchers or teachers, or groups themselves want to see collaborative learning result in. In other words, these are the outcomes of collaboration, which may including individual or team learning, improved social bonds, products created, and so on.

Our simulation procedure focuses on the first three components of Dillenbourg's definition. In particular, the method operationalises the symmetries that define the collaborative situation in terms of systematically different patterns of behaviour, such as those defined by specific roles withing the group. Our operationalisation of interactions and processes is influenced by theories that characterise collaboration an interdependent activity that unfolds in time. When individuals collaborate, they take actions and these actions build upon the prior actions of others—for example, when one team member poses a question and another team member answers. This history of interdependent interactions adds to the common ground (Clark, 1996) that is shared between members of the group. Critically, our understanding of collaboration assumes that the influence this common ground exerts is finite. In other words, those actions that occur within the recent temporal context are the most important for understanding how or why subsequent actions were taken (Ruis et al., 2019).

To understand CPS, researchers collect some record of the group's discourse—that is, a record of the things the group said or did. Typically, these data are labelled for the actor of the discourse—e.g., the speaker—and one or more codes that interpret the discourse in relation to some framework or set of research questions. Each discourse move may be coded monothetically—that is, for one code only—or polythetically—for multiple codes (Graesser et al., 1992). Figure 1 shows an example of this kind of data collected from the virtual internship, *RescuShell*.



Actor	Discourse	Results	Inputs	Outputs	Reasons
1	I had one with aluminium electric with ROM 4 that had 528 payload 290 agility and 6.16 recharge interval its cost was 13115 and safety 191	1	1	1	0
2	Maybe what we should do is agree on which aspect is acceptable to lack.	0	0	0	1
2	Since every design has some component that isn't very satisfactory.	0	0	0	1
3	I think the Pneumatic and Composite would be better than the electric because its very similar in all aspects but a better recharge interval	0	1	1	0
2	I think its ok to lack on recharge interval and cost but the other two are very important I think	0	0	1	1
1	I agree	0	0	0	0

Figure 1: Example of coded collaborative discourse data.

Models that attend to interdependence and temporality measure the frequency with which different actors or the codes describing their actions co-occur within windows of recent temporal context. For example, the window of recent temporal context represented in the Figure 1 above shows the interaction of three speakers and five codes. Popular techniques for analysing CPS and CSCL such as ENA (Shaffer et al., 2016), ONA (Tan et al., 2023), SNA (Moser et al., 2013), LSA(Kapur, 2011), and process mining can be described in these terms.

In sum, our simulation procedure assumes that:

- Groups are composed of two or more actors.
- These actors may hold different roles—or more generally—behave in systematically different ways.
- Collaborative discourse can be represented by sequential records of discourse moves labelled for the actor and coded for content.
- Interactions among actors and among the content of their discourse moves can be modelled via the co-occurrence of actors or codes within windows of moves.

3.2 Procedure Overview

The simulation produces collaborative discourse data of the form shown in Figure X, but without the actual content of the discourse—that is, our method only simulates sequences of speakers and codes.[2] The simulation relies on four key inputs:

An actor matrix A includes the probability of transitioning from one speaker to the next in the discourse data. In cases where impact of speaker transitions is theorised to extend beyond the current discourse move, A is defined as a tensor with separate matrices corresponding to each lag order.

A code matrix $C^{(i)}$ for each actor includes the probability of transitioning from one code to the next in that actor's discourse moves. In cases where impact of speaker transitions is theorised to extend beyond the current discourse move, each $C^{(i)}$ is defined as a tensor with separate matrices corresponding to each lag order.

If the simulated discourse is polythetically coded, a code co-occurrence matrix $O^{(i)}$ includes the probability that codes will co-occur within a single discourse move.

Finally, a propensity matrix P includes the overall probability of an actor using the codes. As described in Appendix A, these matrices can be derived from existing data, created manually, or automatically generated using additional parameters.

[2]As much of the investigation of CPS and CSCL activity occurs at the level of codes, not the discourse itself, this method is justified. However, we discuss this as limitation and opportunity for future work in the Discussion section of the paper.

The simulation procedure includes the following steps for each simulated group activity using the Metropolis-Hastings (MH) (Chib and Greenberg, 1995) variant of Markov Chain Monte Carlo (MCMC) (Gamerman and Lopes, 2006) sampling procedure:

1. Randomly select an actor and code(s) for the first discourse move
2. Select the next speaker using the probabilities in A
3. Propose new code(s) using $O^{(i)}$ and P
4. Compute the acceptance probability of the new code using $C^{(i)}$ and P
5. Accept or reject new code and speaker based on the acceptance probability

See Appendix A for a detailed explanation of the simulation procedure for monothetically and polythetically coded data.

3.3 Research Question 1

To address RQ 1, we used the simulation to reproduce coded discourse from a real CSCL environment. We compared the real and simulated datasets under two conditions—monothetic coding and polythetic coding. As the simulation only produces sequences of speakers and codes, we compared the simulated and real data sets in terms of the co-occurrence structure of the codes. Specifically, we created an ENA model of the real dataset and distributions of ENA models for the simulated datasets, and statistically compared the model of the real data to the distribution of models on the simulated data.

3.3.1 Dataset

Our real data was collected from a single implementation of the virtual internship *Rescushell* at a large university in the Midwestern United States. Virtual internships are online educational experiences where participants act as interns at a fictional company (Chesler et al., 2015). In Rescushell, participants intern at the company *RescuTek* and work together to design and test robotic exoskeletons for rescue workers.

The internship uses a jigsaw approach: each participant is assigned to one of five teams and each team is assigned a unique exoskeleton component on which to base their designs. After exploring the results of using that one component in combination with other kinds of design inputs, the teams are shuffled such that each new team contains at least one person familiar with a particular component. Here, the components are actuators, or the motors used to initiate and control the movement of the exoskeleton. Other inputs include power sources and the kind material used to build the device—e.g., steel or aluminium. Teams test different combinations of inputs and assess the performance of their designs in terms of outputs such as battery life, safety rating, and cost. They judge design performance in relation to standards for these outputs set by internal consultants within the company.

During the internship, participants interacted via the online system *WorkPro*, which includes an integrated email and chat messaging service. All chats were automatically logged by the system. As the chats were logged, they were automatically segmented by the current internship activity and turn of talk. Here, a turn of talk is the text sent when a participant hit return or clicked “send” in the messaging window. Activities were distinct sets of tasks that teams undertook at pre-defined points in the internship. As the focus of our study was collaborative discourse, we only analysed chats from activities that required collaboration, such as design meetings. In total, the observed data includes 1870 turns of talk from 26 individuals. Due to the jigsaw design, each individual participated in two teams for a total of ten teams in the dataset. Each team consisted of five to six individuals.

To address RQ 1a—Can the updated simulation produce *monothetically* coded discourse data with extended representations of common ground?—we used data from Rescushell that had been monothetically coded for eight themes related to engineering design in the context of Resucshell. In particular, the codes: Electric, Series.Elastic, Hydraulic, Pneumatic, and PAM refer to the different actuators that could be included in the designs—i.e., the jigsaw topics. Technical.Constraints refers to other design inputs such as batteries; Performance.Parameters refers to the outputs used to evaluate the designs, such as cost; and Client.and.Consultant.Requests refers to the standards set by the company’s internal consultants (See Table X).

To address RQ 2a—Can the updated simulation method produce *polythetically* coded discourse data with extended representations of common ground?—we used data from Rescushell that had been polythetically coded for six themes. Some of these themes overlapped with those used in RQ 1a including: Technical.Constraints, Performance.Parameters, Client.and.Consultant.Requests. The remaining themes were more general, including: Data—references to prototype evaluation results—Design.Reasoning—references to making particular design or experimental decisions—and Collaboration—references to teamwork (See Table X).

[insert codebooks here]

The codes were applied using automated classifiers. The classifiers used regular expressions to identify the presence or absence of each code in a given turn of talk. The classifiers were validated by comparing their decisions to those of two raters who had achieved sufficient interrater reliability. All raters—two human raters and the classifiers—achieved Cohen’s kappa > 0.90 and Shaffer’s rho < 0.05 for these data (Shaffer and Ruis, 2021).

3.3.2 Simulation

Input matrices and number of discourse moves for each of the ten teams were derived from the real dataset. We simulated the codes above and the possibility of have discourse moves without any codes. Our method produced 1000 sets of simulated discourse moves for each team. Sets from each simulation run were appended to create 1000 separate datasets, each containing the simulated discourse moves of the ten teams to mimic the structure of the real dataset.

To address RQ 1a, we restricted the procedure to produce at most one code per simulated discourse move. We ran the overall simulation procedure four times corresponding to the first four lag orders, or window sizes of two to five discourse moves.

To address RQ 1b, we allowed the procedure to produce simulated discourse moves with up to four codes[3] Again, we ran the overall simulation procedure four times corresponding to the first four lag orders, or window sizes of two to five discourse moves

3.4 Modelling

We used the R implementation ENA—rENA—to create an ENA model for the real dataset and each simulated dataset.

For a given unit of analysis, ENA measures connections between codes that occur within segments of discourse defined by a moving window of fixed length. Connections are defined as the co-occurrence of any pair of codes within the window. This process results in an adjacency matrix for each unit of analysis, where the cells of the matrix are the number of times a given pair of codes co-occurred. The upper triangles of these matrices are converted to adjacency vectors, normalised, and projected into a low dimensional ENA space via singular value decomposition. See Bowman et al. (2021) for a more detailed description of the ENA algorithm.

We used the following ENA parameter values in this study:

- Units of analysis: Individual by team
- Codes: See Table X
- Lines: A discourse move
- Conversations: Discourse moves for each separate team
- Windows: 2-5 lines

3.5 Validation

To validate the simulation results, we compared the observed and simulated data in terms of the similarity between their ENA-derived adjacency vectors. Specifically, we used the method developed by Swiecki 2021 to test whether the observed and simulated adjacency vectors were part of the same distribution. This method calculates the distribution of the average Euclidean distances between the simulated adjacency vectors and their means. This is a distribution of similarity measures between adjacency vectors under the null hypothesis that

[3]Four codes applied to a give discourse move was the maximum present in the real dataset.

they come from the same distribution. Then, the method calculates the average Euclidean distance of the observed adjacency vectors to their corresponding mean simulated vectors. Finally, we compare that average (our test statistic) to the values in the null hypothesis distribution. If the statistic is less than the 95% percentile of the null hypothesis distribution, the test suggests that the observed and simulated vectors are from the same distribution. In other words, *failing to reject the null hypothesis*—i.e., high p-values—suggests that the observed and simulated vectors are very similar, and thus, that the data on which these vectors were generated is very similar.

3.6 Research Question 2

To address RQ 2—What degree of ordering of collaborative discourse is important to account for in models of CPS?—we conducted a sensitivity analysis comparing ENA and ONA models on the same simulated datasets.

ONA extends ENA to account for additional kinds of co-occurrences between codes. Like ENA, ONA produces an adjacency matrix for each unit of analysis in the data; however, in ONA, this matrix is *asymmetric* meaning that it distinguishes between ordered co-occurrences. The matrix also includes counts for co-occurrences of the same code—that is, repeated codes—on the diagonal. As with ENA, these matrices are converted to adjacency vectors, normalised, and projected into a low dimensional space via SVD. The vectors are represented as ONA scores and weighted, directed networks in the low-dimensional space. See Tan et al. (2023) for more details on the ONA algorithm.

Given the differences between ENA and ONA outputs, we augmented the ONA algorithm to produce comparable models. Specifically, we used ONA to create ENA-like models by creating asymmetric adjacency matrices for each unit, zeroing out the diagonals to remove co-occurrences between the same codes, adding the upper and lower triangles, dividing the values by two, and copying these new values back into the matrices. To focus our model comparisons on the effect of the ordering of codes, we also updated the regular ONA outputs to remove co-occurrences between the same codes.

To test the effect of ordering in the data, we systematically introduced asymmetry to the code probability matrices $C^{(i)}$ of each actor. Asymmetry was applied by adjusting the off-diagonal elements of each matrix:

$$C^{(s)}_{k\to l} = C^{(s)}_{k\to l} \times (1 + \eta \times \epsilon), \quad C^{(s)}_{l\to k} = C^{(s)}_{l\to k} \times (1 - \eta \times \epsilon) \tag{1}$$

where η is between 0.1 and 0.9, and ϵ is fixed between 0.9 and 1.1.

In other words, we intentionally pushed the transition probabilities between code pairs in one direction more than the other, using a random asymmetry factor. When $\eta = 0$, $C^{(s)}_{k\to l} \approx C^{(s)}_{l\to k}$. Thus, small values for η tend to produce very symmetric code transition matrices, and large value of η tend to produce very asymmetric code transition matrices.

To conduct the sensitivity analysis, we created a polythetically coded, simulated dataset for each η value at 0.1 intervals. The dataset included five teams, with 1000 discourse moves each; five actors per team; and six simulated codes. To create the input matrices/tensors for each team we sampled parameter values from distributions obtained from the observed Rescushell dataset.

Code transition tensors were initially perfectly symmetric, but were then modified according to the chosen η level. For each simulated dataset, we created and ENA and ONA model using the parameter values above. These models were then compared statistically using t tests between the ENA and ONA scores on a Means-Rotated (MR) dimension (Bowman et al., 2021) which explains the maximal variance between two groups of points. Each test produced a p value and an effect size as measured by Cohen's d. For each set of input parameters and η value this procedure was repeated 1000 times to produce a distribution of p and d values. If 95% of the p values in this distribution were below 0.05, differences between the ENA and ONA models at that η value were considered significant. If 95% of the d values in the distributions were above 0.3, the differences between the ENA and ONA models at that η value were considered meaningful.

4 Results

4.1 Research Question 1

Table 1: Monothetic coding validation for window sizes of 2 to 5.

Window Size	**Test Statistic**	**p-value**	**Confidence Interval**
2	0.61	0.21	(0.54, 0.65)
3	0.66	0.30	(0.60, 0.73)
4	0.56	0.93	(0.61, 0.74)
5	0.66	0.91	(0.70, 0.78)

The results for RQ1a are presented in Table 1. For a window size of two turns, the p-value is 0.21, and the test statistic is 0.61, which falls within the 95% confidence interval of (0.54, 0.65). This suggests that models of the simulated data are statistically similar to models of the observed data. When the window size is extended to three turns, we observed an improvement in the similarity of the models, as suggested by a higher p-value of 0.30. The corresponding test statistic is 0.66, within the confidence interval of (0.60, 0.73). At window sizes of four and five turns, the p-values rise to 0.93 and 0.91, respectively. These high p-values, along with test statistics (0.56 and 0.66)[4] provide strong evidence of the similarity of the simulated and observed models, on average.

The validation results from Table 2 show that for all window sizes, the p-values are 1.00. This indicates that, on average, the observed vectors lie closer

[4]Note that these test statistics fall outside of the corresponding confidence intervals on the lower end, suggesting that the observed model is more similar to the mean simulated model than the individual simulated models.

Table 2: Polythetic coding validation for window sizes of 2 to 5.

Window Size	**Test Statistic**	**p-value**	**Confidence Interval**
2	0.51	1.00	(0.69, 0.77)
3	0.54	1.00	(0.69, 0.78)
4	0.52	1.00	(0.65, 0.74)
5	0.49	1.00	(0.66, 0.75)

to the mean of the simulated vectors than the other simulated vectors do. These consistently high p-values, even at the first order, suggest that allowing multiple codes per turn of talk enhances the alignment between models of the simulated and observed data.

4.2 Research Question 2

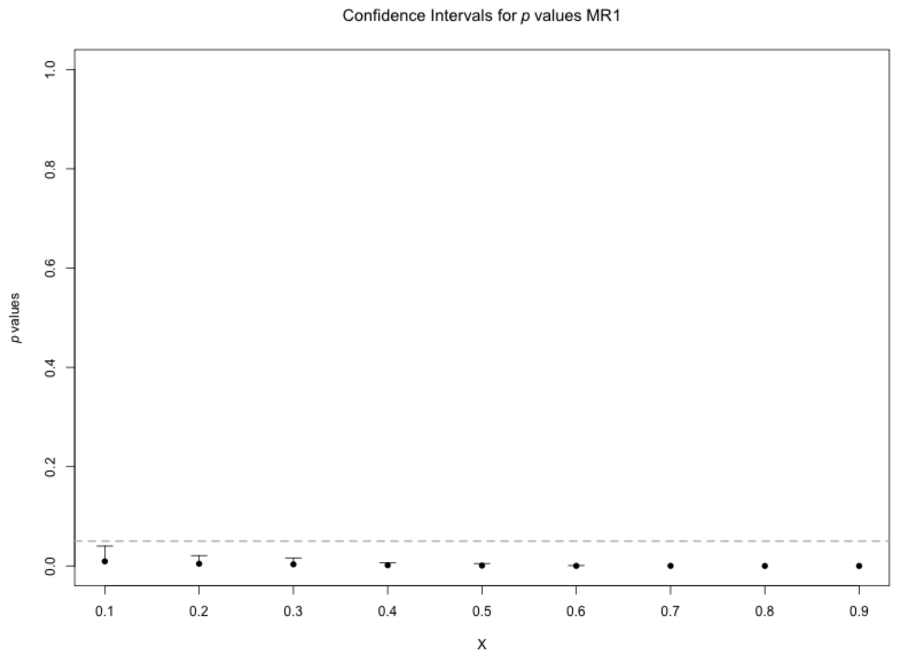


Figure 2: Mean and whisker plot (5%–95% percentile) of p-values for MR1 comparing ENA and ONA across different levels of asymmetry (η), from low to high. Horizontal line at 0.05 indicates the typical significance threshold.

Figures 2 and 3 present the p-values and Cohen's d values, respectively, for comparison between the ENA and ONA scores on the Means Rotated dimension

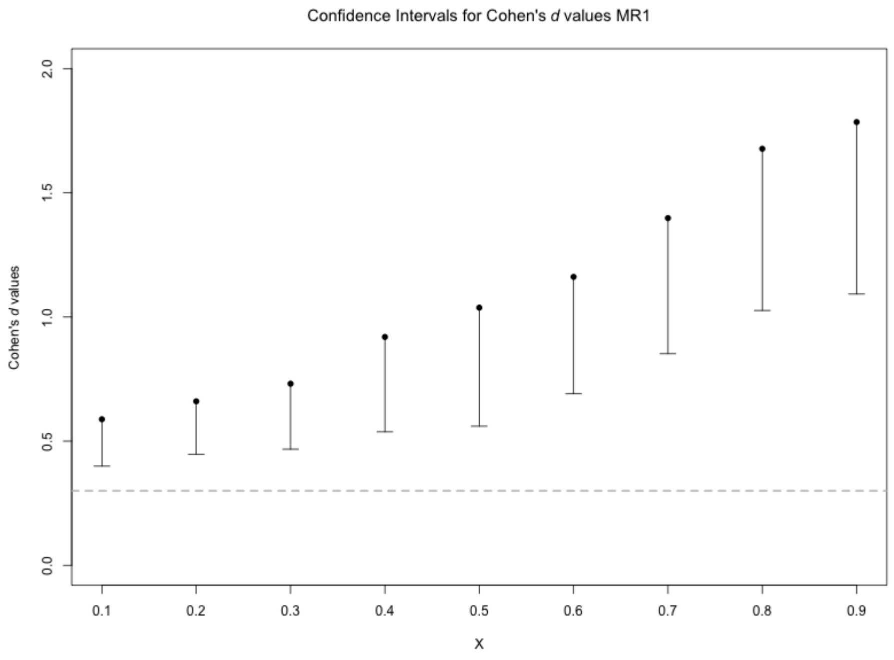


Figure 3: Mean and whisker plot (5%–95% percentile) of Cohen's d values for MR1 comparing ENA and ONA across different levels of asymmetry (η) from low to high. Horizontal line at 0.3 indicates a small-to-medium effect size threshold.

(MR1). In Figure 2, the mean p-values stay well below the 0.05 threshold for all asymmetry levels. The results suggest that even at the lowest asymmetry level ($\eta = 0.1$), there is a statistically significant difference between ENA and ONA models of the same data. Similarly, Figure 3 shows that mean Cohen's d values remain above 0.3 across all asymmetry levels. As asymmetry increases from 0.1 to 0.9, d-values progressively grow, showing that differences between ENA and ONA models get larger as asymmetry increases.

5 Discussion

In this paper, we extended the simulation procedure developed by Swiecki and colleagues (2022) to account for additional features of collaborative discourse—polythetic discourse moves and extended influence of common ground.

Our first research question asked whether the updated method could replicate real collaborative discourse data. To address this question, we sampled parameters from real data as inputs to the simulation procedure. We then applied the same modelling technique—ENA—both the real and simulated datasets. Our results suggest that the simulation procedure can replicate actual collaborative discourse data in terms of the patterns of co-occurrences between discourse codes in the data for monothetically and polythetically coded data with influential common ground operationalised as windows of up to five discourse moves.

Our second research question targeted the utility of the simulation procedure as a method for sensitivity analysis in the context of CSCL and CPS studies. Specifically, we examined how the level of explicit ordering of discourse codes related to model choice. To address this question, we varied the degree to which particular directed co-occurrences between codes were more likely. For each dataset and each simulation run, we compared two widely used models for analysing collaborative discourse data, ENA and ONA. Our goal was to determine the level of ordering that yielded statistically significant differences between ENA/ONA outputs with moderate effect sizes and thus indicated when a model that specifically accounts for order of codes (ONA) is necessary. Our results suggest that when a Means Rotation dimensional reduction is used, very small levels of ordering lead to statistically significant and meaningful differences between models.

Our results have several implications for CSCL and CPS research practice. First, we presented and validated method for simulating a common form of collaborative discourse data. This method can be used to explore a variety of collaborative conditions that go beyond available data. For example, using this simulation method Zhou and colleagues (2025) could have explored the impact of different discussion topics—operationalised as different social and epistemic network configurations—on disagreement. Similarly, Zabolotana and colleagues (2025) could have explored the role that the unobserved phase of collaborative knowledge construction—agree/apply—might have played in distinguishing their two task designs by increasing the prevalence of that code via simulation.

Second, the results suggest that even a small degree of explicit ordering in the data may be enough to warrant the use of a more complex technique like ONA. However, we caution that just because the unordered (ENA) and ordered (ONA) models were statistically significantly different at moderate effect sizes for small levels of asymmetry, this does not necessarily mean that ONA should always be preferred over ENA. Too the contrary, it suggests that researchers should be even more careful when deciding between the two techniques. Small amounts of order across groups in the data may lead to very different results, but care should be taken to examine the data to ensure that order is meaningful to model in relation to specific research questions of interest. Our analysis suggests that common techniques used to analysis collaborative discourse data are very sensitive to the degree of ordering among discourse codes. We interpret these results as meaning that *if you have reason to believe that order is important* you should account for it in the modelling process because it can change the model result drastically. At the same time, if you do not believe that order is important, you should not account for it in the model for the same reason.

Our study is not without limitations. The method cannot provide a means for determining whether factors like the ordering of discourse codes is actually meaningful in part because it only represents collaborative discourse at the level of actors and codes. That is, it does not produce any natural language representation of the discourse that could be used to inform such decisions. Our ongoing work is exploring the combination of this computational simulation, which offers a high degree of control over the outputs but a lack of direct interpretability, with LLM-based simulations that can provide highly-interpretable natural language outputs but lack determinism. This combined computational—LLM simulation paradigm will allow researchers quantitatively test hypotheses while creating a trail of qualitative data that can be used to interpret and validate findings. Additionally, the simulation procedure was validated using only one kind of model, namely ENA. While ENA is well-aligned to features of collaborative discourse because it was developed to account for those features (Shaffer et al., 2016), researchers often use other models for a variety of reasons. The primary assumption of our validation procedure is that collaborative actions are considered interdependent within fixed windows of actions. A variety of models make the same assumption, though they may do so implicitly. Nonetheless, future work should validate the simulation by comparing the results of other models on real and simulated data. Finally, the simulation procedure relies on a small set of assumptions about collaboration and does not accommodate other features they may be important to consider, such the influence of different modalities of discourse (Shaffer et al., 2025). More work is needed to expand the set of assumptions that can be simulated.

6 Conclusion

Despite the limitations above, this work introduces both a method and results that are highly relevant to the CSCL and CPS research communities. Specifically, it describes and validates a procedure for simulating the structure of collaborative discourse under the assumptions that collaboration is best described by interdependent actions that influence one another over fixed periods. Additionally, it illustrates the utility of this method by using the simulation as part of a sensitivity analysis to inform researchers when it may be appropriate to use one widely used analytic technique over another.

Collaboration is a vital skill, and a potentially fruitful context for learning. While real data collected from actual observations of collaborative activity will always be the gold standard, computational simulations offer one way to test hypotheses under conditions beyond what we are readily able to observe. Using such simulations may advance theory and practice in CSCL and CPS research more quickly and with more control, while also suggesting new hypotheses to test under observable conditions, potentially saving time, money, and effort.

A Appendix

A.1 Notation

- Let $A \in \mathbb{R}^{n \times n}$ denote the actor transition matrix, where A_{ij} is the probability of transitioning from actor i to actor j.
- Let $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ be the set of all actors.
- Let $C^{(i)} \in \mathbb{R}^{m \times m}$ denote the code transition matrix for actor A_i, where $C^{(i)}_{kl}$ is the probability that actor A_i transitions from code k to code l in their discourse. This models how actor A_i tends to move between codes across turns.
- Let $O^{(i)} \in \mathbb{R}^{m \times m}$ denote the code co-occurrence matrix for actor A_i, where $O^{(i)}_{kl}$ represents the probability that codes k and l co-occur within a single discourse unit (e.g., turn or utterance). This captures clustering of codes within turns if the discourse is allowed to be polythetically code.
- Let $P \in \mathbb{R}^{n \times m}$ denote the code propensity matrix, where P_{ik} is the overall probability that actor A_i uses code k, estimated as the long-run average frequency of that code across all turns for that actor. This matrix captures differences in baseline tendencies between actors.

Normalisation constraints. All matrices are row-stochastic, meaning the rows sum to 1:

$$\sum_{j=1}^{n} A_{ij} = 1 \quad \forall i, \qquad \sum_{l=1}^{m} C^{(i)}_{kl} = 1 \quad \forall i, k,$$

$$\sum_{l=1}^{m} O^{(i)}_{kl} = 1 \quad \forall i, k, \qquad \sum_{k=1}^{m} P_{ik} = 1 \quad \forall i.$$

Higher-order transitions. To model extended representations of common groups that may include multiple turns of talk for multiple speakers, we generalise transition matrices to tensors. For example, for window sizes of three turns of talk:

- Let $A \in \mathbb{R}^{n \times n \times n}$ be a second-order actor transition tensor, where

$$A_{ijk} = P(\text{next actor} = A_k \mid \text{prev actor} = A_i, \text{current actor} = A_j).$$

- Let $C^{(i)} \in \mathbb{R}^{m \times m \times m}$ be a second-order code transition tensor for actor A_i, where

$$C^{(i)}_{klm} = P(\text{next code} = c_m \mid \text{prev code} = c_k, \text{current code} = c_l, \text{actor} = A_i).$$

All conditional slices are normalised so that $\sum_k A_{ijk} = 1$ and $\sum_m C^{(i)}_{klm} = 1$.

A.2 Monothetic Coding Simulation

To simulate monothetically coded discourse, we use a Markov Chain Monte Carlo (MCMC) method combined with the Metropolis–Hastings (MH) algorithm. The MCMC is used to approximate samples from a probability distribution that is difficult to draw from directly. It does this by constructing a sequence of samples where the long-run, or stationary, behaviour of the chain matches the desired distribution. At each step, a candidate sample is generated from a simpler proposal distribution. The candidate is accepted with a probability based on the ratio of target densities—that is, how likely the candidate is under the distribution of interest compared to the current sample. If accepted, the chain moves to the candidate; if rejected, it stays at the current sample. Repeating this process many times produces a chain of samples that that can be treated as approximate draws from the target distribution.

The MCMC simulation proceeds as follows:

1. **Propose next actor**: The next actor is sampled from the actor transition tensor A, conditional on the recent actor history.

2. **Propose next code**: Given the proposed actor A_i, a candidate code c' is proposed via uniform sampling from the set of codes excluding the current one:
$$P(c') = \frac{1}{m-1}, \quad c' \neq c.$$

3. **Acceptance probability**: The MH acceptance ratio α compares the likelihood of the proposed vs. current state, combining (a) the actor-specific code propensity P_{ik} and (b) the actor-specific code transition probabilities $C^{(i)}$. For example, for a window size of two turns:
$$\alpha = \min\left(1, \frac{P_{ic'} \cdot C^{(i)}_{c_{t-1},c'}}{P_{ic} \cdot C^{(i)}_{c_{t-1},c}}\right).$$

For higher-order models, the transition terms generalise to condition on longer code histories.

4. **Accept or reject**: A uniform random draw $u \sim U(0,1)$ determines whether the proposal is accepted ($u < \alpha$) or the chain remains in the current state.

The acceptance probability compares the plausibility of the proposed code to that of the current code, given both the actor's overall code usage tendencies and their sequential code dynamics. The numerator of the ratio, $P_{ic'} \cdot C^{(i)}_{c_{t-1},c'}$, represents the likelihood that actor A_i would produce the proposed code c', combining the actor's baseline propensity to use that code ($P_{ic'}$) with the probability of transitioning to it from their previous code ($C^{(i)}_{c_{t-1},c'}$). The denominator, $P_{ic} \cdot C^{(i)}_{c_{t-1},c}$, reflects the same joint likelihood for the actor's current code c. Taking the ratio of these two likelihoods expresses how much more consistent the proposed code is with the actor's behaviour compared to the current one. If the proposed code is more probable, the ratio exceeds one and the move is always accepted; if it is less probable, the ratio falls between zero and one and the move is accepted only with that probability. The use of the $\min(1, \cdot)$ function ensures that the acceptance probability never exceeds one.

A.3 Polythetic Coding

A.3.1 Number of Codes

To simulate turns of talk in which multiple codes may co-occur, we extend the monothetic procedure by introducing a Poisson process to model the number of codes per turn, together with a modified proposal function and acceptance ratio.

The number of codes in a turn is modeled using a Poisson distribution with parameter λ, representing the mean number of codes per turn. A global rate parameter, λ_g, is estimated from observed data by averaging the number of codes across all N turns:

$$\lambda_g = \frac{1}{N} \sum_{i=1}^{N} x_i, \tag{2}$$

where x_i is the number of codes in turn i. This provides a baseline rate of code occurrence across the discourse.

The proposal function in the Metropolis–Hastings algorithm is responsible for suggesting the prospective set of codes for each turn. This involves two steps: determining the number of codes and selecting the specific codes.

To adapt the Poisson process to local conversational context, we compute a turn-specific rate parameter λ_l by modulating the global rate λ_g with cross-turn code co-occurrence probabilities. For this purpose, we introduce a group-level cross-turn co-occurrence matrix $Q \in \mathbb{R}^{m \times m}$, where

$Q_{ji} = P(\text{code } i \text{ appears in turn } t \mid \text{code } j \text{ appeared in one of the previous } k \text{ turns}).$

The local Poisson rate is then given by:

$$\lambda_l = \lambda_g \times \left(\sum_{i \in C_t} \sum_{j \in C_{t-k:t-1}} Q_{ji} \right), \tag{3}$$

where C_t is the set of candidate codes for the current turn, and $C_{t-k:t-1}$ is the union of codes across the previous k turns. This adjustment ensures that the expected number of codes is sensitive to recent discourse context. Essentially, this process computes a weighted sum of how strongly the codes in the current turn are associated with codes from the recent past. If many of the candidate codes frequently co-occurred with recent codes, λ_l gets larger and the model expects more codes in the next line. If few or none co-occur, λ_l stays small and the model expects fewer codes. The number of codes for the current turn is then sampled from a Poisson distribution with parameter λ_l

A.3.2 Code Proposal

Once the number of codes is determined, the specific codes are proposed. To balance topic persistence with variability, we apply an exclusion probability of 0.5 to each code from the previous turn. Codes retained after this step form part of the candidate pool, while others are excluded to encourage exploration of new topics.

Let A_i be the current actor for turn t. After applying the exclusion step, let J be the pool of available codes and let $\hat{x}$ be the number of codes to propose for this turn (drawn from the Poisson process described earlier). We propose an *ordered* tuple $(c'_1, \ldots, c'_{\hat{x}})$ by sampling one code at a time, without replacement, with probabilities proportional to (i) the actor-specific code propensity $P_{i\cdot}$ and (ii) a within-turn co-occurrence weight derived from $O^{(i)}$ and the codes already chosen for this turn.

For a candidate code c' and a current partial set $S = \{c'_1, \ldots, c'_{r-1}\}$, define

$$q(c' \mid S) \;=\; \prod_{u \in S} O^{(i)}_{u,c'}\,, \qquad \text{with the convention } q(c' \mid \varnothing) = 1.$$

At step r ($1 \le r \le \hat{x}$), with $S_{r-1} = \{c'_1, \ldots, c'_{r-1}\}$ and remaining pool $J_{r-1} = J \setminus S_{r-1}$, select c'_r with probability

$$\Pr(c'_r = j \mid S_{r-1}) \;=\; \frac{P_{ij}\; q(j \mid S_{r-1})}{\displaystyle\sum_{k \in J_{r-1}} P_{ik}\; q(k \mid S_{r-1})}\,, \qquad j \in J_{r-1}.$$

The proposal probability of the ordered tuple $(c'_1, \ldots, c'_{\hat{x}})$ is the product of the step-wise conditionals:

$$P_{\text{prop}}\big(c'_1, \ldots, c'_{\hat{x}}\big) \;=\; \prod_{r=1}^{\hat{x}} \frac{P_{i\,c'_r} \displaystyle\prod_{u \in S_{r-1}} O^{(i)}_{u,c'_r}}{\displaystyle\sum_{k \in J_{r-1}} P_{ik} \prod_{u \in S_{r-1}} O^{(i)}_{u,k}}\,, \quad S_{r-1} = \{c'_1, \ldots, c'_{r-1}\},\; J_{r-1} = J \backslash S_{r-1}.$$

At the first pick ($S_0 = \varnothing$) the co-occurrence term is $q(\cdot \mid \varnothing) = 1$, so selection is driven purely by the actor's baseline propensities $P_{i\cdot}$. Thereafter, each candidate's weight is up- or down-weighted by how well it co-occurs (in $O^{(i)}$) with the codes already chosen for *this* turn, producing cohesive within-turn bundles. The outer product across r yields the joint probability of proposing the entire ordered tuple via sequential, without-replacement draws.

A.3.3 Acceptance probability

Next, the we calculate the acceptance ratio for the proposed code set (above) and the likelihood of the target distribution. The target distribution encodes how plausible a set of codes C is, given the actor's baseline tendencies and the sequential context. For actor A_i in turn t, we define the normalised likelihood of a set of codes C_t as

For actor A_i, the unnormalised likelihood of a candidate code set C_t is defined as

$$L(C_t) \;\propto\; \prod_{c \in C_t} P_{ic} \;\times\; \prod_{c \in C_t} C^{(i)}_{c_{t-p},\ldots,c_{t-1},\,c}, \tag{4}$$

where P_{ic} is the actor's baseline propensity for code c, and $C^{(i)}_{c_{t-p},\ldots,c_{t-1},\,c}$ is the probability of transitioning from the previous p codes $(c_{t-p}, \ldots, c_{t-1})$ into code c, as determined by the actor-specific code transition tensor $C^{(i)}$.

The MH acceptance probability compares the plausibility of the proposed set C' to the current set C_t under the target distribution:

$$\alpha = \min\!\left(1,\; \frac{L(C')}{L(C_t)}\right)$$

The likelihood ratio $L(C')/L(C_t)$ evaluates whether the proposed set is more consistent with the actor's propensities and recent history than the current set.

Intuitively, if the proposed set is more plausible, α exceeds one and the move is always accepted; if it is less plausible, α falls between zero and one and the move is accepted with that probability.

Finally, the algorithm decides whether to adopt the proposed set C' or retain the current set C_t. A uniform random variable $u \sim U(0,1)$ is drawn; if $u < \alpha$, the proposal is accepted and C_t is updated to C'. Otherwise, the proposal is rejected and the chain remains in its current state. This mechanism ensures that moves to more plausible states are always accepted, while moves to less plausible states are still occasionally accepted, allowing the chain to explore the state space and avoid becoming trapped in local modes.

The acceptance probability in the polythetic case extends the logic of the monothetic simulation from single codes to sets of codes. It evaluates whether the entire bundle of proposed codes is more or less consistent with the actor's typical behaviour and the sequential dynamics of the discourse than the current bundle. Sets that align well with both the actor's overall propensities and the recent code history are usually accepted, while less consistent sets are less accepted.

A.4 Generation of Simulation Components

All matrices described above can be calculated from data of the form shown in Figure X. For sensitivity analysis, additional parameters can be use to used to generate the matrices from scratch and systematically manipulate their structure.

A.4.1 Actor Transition Tensor A

1. **Initialisation**: Fill with random positive values.

2. **Asymmetry adjustment**: Optionally, to test the effect of speaker ordering, adjust off-diagonal elements:

   $$A_{i\to j} = A_{i\to j} \times (1 + \eta \times \epsilon), \quad A_{j\to i} = A_{j\to i} \times (1 - \eta \times \epsilon),$$

   where $\eta \in [0.1, 0.9]$ and $\epsilon \in [0.9, 1.1]$.

3. **Diagonal dominance**: To control persistence—or likelihood of repeated sequences of the same actor—we apply a diagonal dominance parameter $\delta > 0$. For each row of the transition matrices, the diagonal entry is inflated relative to the off-diagonal entries:

   $$A_{ii} \leftarrow A_{ii} \times (1 + \delta),$$

   - When $\delta = 0$, persistence is determined only by the initial random draw and asymmetry adjustments.
   - Larger values of δ increase the likelihood that actors repeat consecutively, producing long runs of the same state.
   - Smaller values of δ lead to more frequent switching between states.

4. **Normalisation**: Scale each row or conditional slice to sum to 1.

A.4.2 Code Transition Tensor $C^{(i)}$

1. **Initialisation**: For each actor A_s, fill $C^{(s)}$ with random draws.
2. **Asymmetry adjustment**: Optionally, to test the effect of code ordering
$$C^{(s)}_{k\to l} = C^{(s)}_{k\to l} \times (1 + \eta \times \epsilon), \quad C^{(s)}_{l\to k} = C^{(s)}_{l\to k} \times (1 - \eta \times \epsilon),$$
where $\eta \in [0.1, 0.9]$ and $\epsilon \in [0.9, 1.1]$.
3. **Diagonal dominance**: As with the actor transition matrices, we control persistence—or likelihood of repeated sequences of the same code—by applied apply a diagonal dominance parameter $\delta > 0$. For each row of the transition matrices, the diagonal entry is inflated relative to the off-diagonal entries:
$$C_{ii} \leftarrow C_{ii} \times (1 + \delta),$$
4. **Sparsity control**: Optionally, set a fraction of entries to near-zero.
5. **Normalisation**: Ensure each conditional slice sums to 1.

A.4.3 Code Usage Propensity P

1. **Initialisation**: For each actor, draw a probability vector of length m from a Dirichlet distribution with concentration parameter α. The Dirichlet distribution is a natural choice for generating valid probability vectors because it ensures all entries are positive and sum to one.
2. **Interpretation**: The parameter α controls the "evenness" of the distribution. Small values of α produce highly skewed vectors, where one or two codes dominate (specialist actors). Large values of α yield nearly uniform distributions, where all codes are used with roughly equal frequency (generalist actors).
3. **Normalisation**: The Dirichlet distribution is already normalized; each row of P automatically sums to 1.

A.4.4 Within-Turn Co-occurrence Matrix $O^{(i)}$

1. **Initialisation**: For each actor, generate a symmetric matrix of positive random values. Set the diagonal to zero to disallow a code from co-occurring with itself.
2. **Distribution shaping**: Apply simple transformations to shape the co-occurrence structure. In particular:
    - *Skewness adjustment*: biases the matrix so that a small subset of code pairs have much higher probabilities than others, producing "preferred" combinations.

- *Kurtosis adjustment*: controls whether the distribution is very peaked (one or two pairs dominate, high kurtosis) or flatter (many pairs moderately likely, low kurtosis). A simple transformation is

$$O_{kl} \leftarrow (O_{kl})^{\gamma},$$

where $\gamma > 1$ sharpens the distribution (increasing kurtosis) and $0 < \gamma < 1$ flattens it (decreasing kurtosis).

3. **Normalisation**: After shaping, each row is scaled to sum to 1. This ensures that $O^{(i)}$ defines a proper conditional probability distribution.

References

Aronson, E. (1978). *The jigsaw classroom.* Sage, Newbury Park, CA. Pages: 197.

Baker, M. and Reimann, P. (2025). Editorial notes: Core issues in CSCL research. *International Journal of Computer-Supported Collaborative Learning*, 20(2):165–170.

Bergner, Y., Andrews, J. J., Zhu, M., and Gonzales, J. E. (2016). Agent-Based Modeling of Collaborative Problem Solving. *ETS Research Report Series*, 2016(2):1–14.

Bowman, D., Swiecki, Z., Cai, Z., Wang, Y., Eagan, B., Linderoth, J., and Shaffer, D. W. (2021). The Mathematical Foundations of Epistemic Network Analysis. In Ruis, A. R. and Lee, S. B., editors, *Advances in Quantitative Ethnography*, pages 91–105, Cham. Springer International Publishing.

Chesler, N. C., Ruis, A. R., Collier, W., Swiecki, Z., Arastoopour, G., and Williamson Shaffer, D. (2015). A Novel Paradigm for Engineering Education: Virtual Internships With Individualized Mentoring and Assessment of Engineering Thinking. *Journal of Biomechanical Engineering*, 137(2).

Chi, M. T. (1997). Quantifying qualitative analyses of verbal data: A practical guide. *The journal of the learning sciences*, 6(3):271–315.

Chib, S. and Greenberg, E. (1995). Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49(4):327–335. Publisher: ASA Website _eprint: https://www.tandfonline.com/doi/pdf/10.1080/00031305.1995.10476177.

Christopher Frey, H. and Patil, S. R. (2002). Identification and Review of Sensitivity Analysis Methods. *Risk Analysis*, 22(3):553–578. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/0272-4332.00039.

Clark, H. H. (1996). *Using language.* Cambridge university press.

Dillenbourg, P. (1999). *Collaborative learning: Cognitive and computational approaches. advances in learning and instruction series.* ERIC.

Gamerman, D. and Lopes, H. F. (2006). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference, Second Edition.* Chapman and Hall/CRC, New York, 2 edition.

Gilber, N. and Troitzsch, K. (2005). *Simulation for the social scientist.* McGraw-Hill Education, UK.

Graesser, A. C., Person, N., and Huber, J. (1992). Mechanisms that Generate Questions. In *Questions and Information Systems.* Psychology Press. Num Pages: 22.

Halpin, P. F. and von Davier, A. A. (2017). Modeling Collaboration Using Point Process. In *Innovative Assessment of Collaboration*, pages 233–247. Springer.

Hesse, F., Care, E., Buder, J., Sassenberg, K., and Griffin, P. (2015). A framework for teachable collaborative problem solving skills. In *In P. Griffin & E. Care (Eds.), Assessment and teaching of 21st century skills: Methods and approach*, pages 37–56. Springer, Dordrecht, the Netherlands.

Hutchins, E. (1995). *Cognition in the wild.* MIT Press, Cambridge, MA.

Jager, W., Popping, R., and Van de Sande, H. (2001). Jager, W., Popping, R., & Van de Sande, H. (2001). Clustering and fighting in two-party crowds: Simulating the approach-avoidance conflict. *Journal of Artificial Societies and Social Simulation*, 4(3):1–18.

Kapur, M. (2011). Temporality matters: Advancing a method for analyzing problem-solving processes in a computer-supported collaborative environment. *International Journal of Computer-Supported Collaborative Learning*, 6(1):39–56.

Martinez-Maldonado, R., Gašević, D., Echeverria, V., Nieto, G. F., Swiecki, Z., and Shum, S. B. (2021). What Do You Mean by Collaboration Analytics? A Conceptual Model. *Journal of Learning Analytics*, 8(1):126–153.

Moser, C., Groenewegen, P., and Huysman, M. (2013). Extending Social Network Analysis with Discourse Analysis: Combining Relational with Interpretive Data. In Özyer, T., Rokne, J., Wagner, G., and Reuser, A. H., editors, *The Influence of Technology on Social Network Analysis and Mining*, volume 6, pages 547–561. Springer Vienna, Vienna. Series Title: Lecture Notes in Social Networks.

Reimann, P., Frerejean, J., and Thompson, K. (2009). Using process mining to identify models of group decision making in chat data. In *Proceedings of the 9th international conference on Computer supported collaborative learning-Volume 1*, pages 98–107. International Society of the Learning Sciences.

Ruis, A. R., Siebert-Evenstone, A. L., Pozen, R., Eagan, B. R., and Shaffer, D. W. (2019). Finding common ground: A method for measuring recent temporal context in analyses of complex, collaborative thinking. In *13th International Conference on Computer-Supported Collaborative Learning*.

Shaffer, D., Wang, Y., and Ruis, A. (2025). Transmodal Analysis. *Journal of Learning Analytics*, pages 1–22.

Shaffer, D. W., Collier, W., and Ruis, A. R. (2016). A tutorial on epistemic network analysis: Analyzing the structure of connections in cognitive, social, and interaction data. *Journal of Learning Analytics*, 3(3):9–45.

Shaffer, D. W. and Ruis, A. R. (2021). How We Code. In Ruis, A. R. and Lee, S. B., editors, *Advances in Quantitative Ethnography*, pages 62–77, Cham. Springer International Publishing.

Swiecki, Z. (2021). The expected value test: A new statistical warrant for theoretical saturation. In *Advances in Quantitative Ethnography: Third International Conference, ICQE 2021, Malibu, CA, USA, November 6-11, Proceedings.*

Swiecki, Z., Lian, Z., Ruis, A. R., and Shaffer, D. W. (2019). Does order matter? Investigating sequential and cotemporal modesl of collaboration. In Lund, K., Niccolai, G. P., Lavoué, E., Hmelo-Silver, C., Gweon, G., and Baker, M., editors, *A Wide Lens: Combining Embodied, Enactive, Extended, and Embedded Learning in Collaborative Settings, 13th International Conference on Computer-Supported Collaborative Learning (CSCL) 2019*, volume 1, pages 112–120, Lyon, France. International Society of the Learning Sciences.

Swiecki, Z., Marquart, C., and Eagan, B. (2022). Simulating Collaborative Discourse Data. In *Proceedings of the 15th International Conferences on Computer-Supported Collaborative Learning - CSCL 2022*, pages 83–91, Hiroshim, Japan. International Society of the Learning Sciences.

Tan, Y., Ruis, A. R., Marquart, C., Cai, Z., Knowles, M. A., and Shaffer, D. W. (2023). Ordered Network Analysis. In Damşa, C. and Barany, A., editors, *Advances in Quantitative Ethnography*, volume 1785, pages 101–116. Springer Nature Switzerland, Cham. Series Title: Communications in Computer and Information Science.

Tisue, S. and Wilensky, U. (2004). NetLogo: A Simple Environment for Modeling Complexity. *Paper presented at the International Conference on Complex Systems.*

Zabolotna, K., Nøhr, L., Iwata, M., Spikol, D., Malmberg, J., and Järvenoja, H. (2025). How does collaborative task design shape collaborative knowledge construction and group-level regulation of learning? A study of secondary school students' interactions in two varied tasks. *International Journal of Computer-Supported Collaborative Learning*, 20(2):171–199.

Zhou, J., Hmelo-Silver, C. E., Ryan, Z., Stiso, C., Murphy, D., Danish, J., Chinn, C. A., and Duncan, R. G. (2025). Disagreeing softly: Supporting students in managing disagreement in peer critique. *International Journal of Computer-Supported Collaborative Learning*, 20(2):249–282.

Zöller, N., Morgan, J. H., and Schröder, T. (2021). Modeling Interaction in Collaborative Groups: Affect Control Within Social Structure. *Journal of Artificial Societies and Social Simulation*, 24(4):6.