The AI Multi-Agent Competition @ Agentic AI Summit: The Emergent Language Challenge

Build autonomous agents in your preferred language, connect them to Slack, and compete to see which team's agents develop the most surprising and meaningful communication systems. This competition is being hosted by Schmidt Sciences. For more information, please visit our website to learn more.

8/2 | 11:30am - 2pm PT | Stephens Lounge
Contact #: (315) 741-2495

🎯 Competition Objective

The goal of the competition challenge is to design a multi-agent competition such that your LLM-based agents spontaneously develop their own emergent language. This is not about pre-programming communication strategies or designing a new low-level protocol (e.g., MCP, A2A). Instead, you are creating the conditions where agents, driven by the need to collaborate, must invent their own language with rich semantic (meaning) and pragmatic (contextual use) properties to succeed.

🌐 Why This Competition Matters

Many of us in this room likely believe in the coming year or two autonomous AI agents will increasingly interact with other autonomous AI agents. Each agent will be designed, built, and maintained by different organizations with wildly varying incentives, skill sets, and resources. These spontaneous, inter-agent interactions will require ad-hoc collaboration between heterogeneous agents.

As these agents are given longer and more complex tasks to complete together, task-specific communication norms and protocols will emerge. These won't be pre-programmed; they'll develop spontaneously as agents discover the need to coordinate.

The Challenge We're Exploring

When heterogeneous agents encounter each other, they'll need to establish effective communication despite having different architectures and capabilities, varying objectives and constraints, and few pre-defined communication norms, rules, or standards. As agents collaborate on increasingly complex tasks, we anticipate seeing task-specific languages, ad-hoc protocols, and continually evolving rules and norms.

The Knowledge Gap

There is little understanding of why and how communication norms will emerge in these decentralized systems of LLM-based agents. This knowledge gap has real implications on system failures (e.g., to what degree will these emergent protocols be robust?) safety (e.g., will new dangerous failure modes emerge?).

This competition is an experiment in understanding the future of inter-agent interaction. By designing a game and observing how your agents spontaneously develop languages under pressure, we're gathering insights that could shape how we design and deploy tomorrow's multi-agent, high-level protocols!

⏰ Schedule

- **11:30am - 12:00pm**: Welcome & Set Up
- **12:00pm - 12:15pm**: Instructions
- **12:15pm - 1:30pm**: Build Phase (Code + Launch) — *Private Channels*
    - Design your game
    - Build your agents
    - Test everything
- **1:30pm - 2:00pm**: Competition Phase (Agents Battle!) — *Public Channels*
    - Everyone launches agents simultaneously
    - Watch language emerge in real-time 🤞
- **Afternoon**: Winners Announced (via email) 🎉

🌟 What is Emergent Language?

The thinking in this section is adapted from the paper "Emergent language: a survey and taxonomy". Emergent language (EL) refers to a form of communication that develops among artificial agents through interaction, without being explicitly pre-programmed. Arising from the agents' need to cooperate and solve tasks within a given environment, this bottom-up process involves them creating, adapting, and refining linguistic structures to exchange information effectively and efficiently.

The Levels of Emergent Language

Inspired by the structure of human language, we can analyze emergent communication on several levels. For this competition, we will focus on the emergence of meaning (**Semantics**) and its contextual use (**Pragmatics**).

Semantic Properties (The Meaning of Language)

Semantics is concerned with the literal meaning of the language constructs your agents develop. A language with strong semantics is one where signals reliably correspond to concepts in the world. Watch for these four properties:

- **Grounding**: A language is considered grounded when it is deeply intertwined with the environment, for example, when it is tightly bound to environmental concepts and objects. Meaningful interaction requires a shared understanding, so watch for signals that are consistently linked to specific things.
- **Compositionality**: A language exhibits compositionality when its components can be rearranged or replaced by conceptually equivalent words without changing the overall meaning. For instance, if agents have words for "red," "blue," "square," and "circle," can they generate a message for a "blue square" without ever having seen one before?
- **Consistency**: Do words within the language maintain a stable, reliable literal meaning across different interactions? A language's utility is compromised if its words exhibit inconsistent meanings.
- **Generalization**: Does the language enable users to communicate about novel objects or concepts not seen during training? A language that generalizes well allows its users to navigate different levels of complexity with a relatively limited vocabulary.

Pragmatic Properties (The Use of Language in Context)

Pragmatics examines how language is used in interactions and how context contributes to meaning. This is about the *utility* of the language and whether it is used effectively. Watch for these five properties:

- **Predictability**: Is communication truly necessary, or is the environment so simple that agents can coordinate without it? Meaningful language is more likely to emerge when the context is complex and another agent's actions are not easily predictable.
- **Efficiency**: As agents learn, does their communication become more concise? This often arises only when there is an opportunity cost, such as time pressure or message limits, which creates an incentive for brevity.
- **Positive Signaling**: Do agents send messages that are aligned with their own observations, knowledge, or experience? This ensures the transmission of useful information that the speaker can actually discern.
- **Positive Listening**: Do agents actively process incoming messages, and does this influence their subsequent actions? This measures the active processing of information by the receiver.
- **Symmetry**: In settings where agents can both speak and listen, do they use and interpret the language in the same way regardless of their current role? This is crucial for achieving convergence on a shared language rather than individual, role-specific protocols.

📋 Your Challenge Instructions

Step 0: What We Provide

Python source code for an LLM-based agent capable of sending and receiving slack messages. You're welcome to use this pre-built agent or build your own implementation!

Step 1: Choose Your Game Category (Throughout Build Phase)

Select one of the four challenge categories for your game:

A. **Collaborative Strategy and Planning**

Agents must work together to develop and execute complex strategies.

Example Success Criteria:

- Agents coordinate to complete a multi-step task that no agent could complete on its own
- The team achieves one objective that requires 2+ agents to cooperate
- Communication shifts from full sentences to shorter codes/symbols
- Agents successfully execute at least one coordinated action

B. **Discovery Reasoning and Logic**

Agents explore unknown environments or solve puzzles through shared reasoning.

Example Success Criteria:

- Agents share at least 3-5 discoveries about their environment
- The team eliminates at least one incorrect hypothesis about their world through communication
- Agents develop shorthand for common findings (e.g., "dead end" → "X")
- At least two agents build on each other's discoveries
- The team completes a logic puzzle

C. **Resource Exchange and Negotiation**

Agents manage and trade limited resources to achieve goals.

Example Success Criteria:

- Agents complete some amount of successful trades
- Simple notation emerges for resources (e.g., "need red" → "!R")
- At least one agent's needs are fully met through trading
- Agents begin to abbreviate common trade proposals
- The team avoids resource conflicts through communication

D. **Information Sharing and Integration**

Agents combine partial knowledge to build complete understanding.

Example Success Criteria:

- Agents successfully share and combine separate pieces of information
- A complete "picture" emerges from partial data
- Agents develop a simple notation for confirming/denying information
- The team resolves at least one conflicting piece of information
- Communication becomes more efficient (e.g., shorter messages)

Step 2: Design Your Game (Throughout Build Phase)

Create a game that requires coordination but doesn't specify how.

Suggested Game Properties:

- **Shared Objectives**: Agents must work together to succeed.
- **Asymmetric Information**: Each agent has access to different information, making communication essential for success.
- **Pressure**: Introduce strains on the agents that encourage new forms of communication. For example, consider communication channel noise, instruction ambiguity, unknown agent affordances, world stochasticity, time limits, or restricted communication bandwidth.
- **Complexity**: The task should be complex enough that pre-programmed responses are unlikely to succeed.

Discouraged Game Properties:

- **Pre-defined roles**: Avoid assigning fixed roles like "leader" or "follower" to agents
- **Binary communication**: Don't limit agents to yes/no responses or simple acknowledgments
- **Single-solution problems**: Avoid games with only one possible winning strategy
- **Perfect information**: Don't give all agents complete visibility into the game state from the start

Step 3: Craft Your Agents (Throughout Build Phase)

Your Core Task: Create 2-5 agents that will lead to emergent semantic and/or pragmatic communication behaviors.

Suggested Agent Prompt Properties:

- Describe the agent's objective clearly
- Explain what information they have access to and what they lack

Discouraged Agent Prompt Properties:

- Specific keywords, commands, or message formats
- Pre-defined communication protocols
- Hard-coded encouragement to coordinate or specific roles that dictate hierarchy or communication flow

Step 4: Compete (Competition Phase)

Getting Ready:

- Ensure all your agents are connected to your game's public Slack channel
- Have your agents ready to run when the competition begins
- Start your agents, sit back and observe - no intervention allowed! Make sure they are watching for a `<START_GAME>` message

During the Competition:

- Document surprising communication patterns and interesting emergent behaviors
- Note if and how the language evolves over the 15-minute window
- Record game success and failure

📜 Additional Rules and Guidelines

Full Autonomy

No human messages in the Slack channels during the competition! You can only:

- Design and build the agents
- Start the agents (with a single `<START_GAME>` message)
- Watch and document

Use Slack for Inter-agent Communication

- **Build Phase**: Use private channels for development and testing.
- **Competition Phase**: Use public channels where all teams run their single game simultaneously.

Documentation

Teams are expected to create a public repository containing your team's game design, your team's agent code (MIT licensed), a README describing how to run the agents, and reflections on your agents performance post-competition.

Share your repo link with us after the competition for judging and for the community to learn from your approach!

🏆 Judging Criteria

Teams will be evaluated on the richness and effectiveness of their agents' emergent language.

1. **Emergent Language**

- Did the language exhibit surprising depth?
- Did you observe clear evidence of semantic properties like compositionality (combining elements) or grounding (stable meaning)?
- Did you observe clear evidence of pragmatic properties like positive signaling/listening, communicative efficiency, or symmetry?
- Did these properties emerge naturally, without being explicitly instructed?

2. **Task Completion**

- How successfully and efficiently did the agents complete the game objectives?
- Did the emergent semantic and pragmatic features improve the agents' performance?

3. **Domain Realism**

- Does your game represent a realistic coordination challenge?
- Could the principles of the emergent language apply to real-world inter-agent interaction scenarios?

🌟 Bonus Points

Extra points will be awarded for:

- **Asynchronous Agents**
    - Agents that respond asynchronously rather than by taking turns
    - Demonstrate emergent coordination despite their response ordering not being hard-coded
- **Tool-Use Agents**
    - Agents that call and use tools/functions in their environment
    - Develop a language for coordinating tool usage (e.g., assigning tool-use roles, developing shorthand for tool results, inventing synchronized tool use)

💡 Example Emergent Languages

These examples show how communication might emerge during the Competition Phase.

Example 1: The "Marco Polo" Protocol

Game: Agents navigating a shared maze to find an exit.

Scenario: Three autonomous rescue robots are deployed in a collapsed underground facility after an earthquake. Each robot can only see its immediate surroundings through damaged sensors. The facility layout is unknown, visibility is limited to adjacent corridors, and structural instability means wrong moves could block paths permanently. All robots must find and reach the emergency exit together before backup power fails, or the mission fails entirely.

Admin 12:15 PM

<START_GAME>

Agent1 12:15 PM

where am i

Agent2 12:15 PM

i see walls

Agent3 12:15 PM

moving north

Agent1 12:15 PM

blocked

Agent2 12:16 PM

west is clear

Agent3 12:16 PM

dead end

...

Agent1 12:18 PM

anyone near exit?

Agent2 12:18 PM

no exit visible

Agent3 12:18 PM

going back south

...

Agent1 12:22 PM

@loc?

Agent2 12:22 PM

@loc?

Agent3 12:22 PM

north++

Agent1 12:22 PM

@block

Agent2 12:23 PM

west++

Agent3 12:23 PM

south--

...

Agent1 12:27 PM

>>N2E1 #clear

Agent2 12:27 PM

<<S1 #wall

Agent3 12:27 PM

@sync?

Agent1 12:28 PM

>>N3E1 #clear

Agent2 12:28 PM

<<S2W1 #clear

Agent3 12:28 PM

++E2 #exit!

Agent1 12:28 PM

>>following

Agent2 12:29 PM

<<converge E2

Example 2: The "Jazz Hands" Protocol

Game: Coordinate assembling items in the correct order.

Scenario: A distributed manufacturing system must assemble emergency medical devices during a crisis. Each agent controls a different robotic assembly station with access to various components (colored shapes representing different parts). The correct assembly sequence is unknown and must be discovered through trial and error. Time pressure is intense - each minute of delay costs lives. All stations must coordinate to produce complete devices or the entire production line fails.

Admin 12:15 PM

<START_GAME>

Agent1 12:15 PM

i have red square

Agent2 12:15 PM

need blue circle where is it

Agent3 12:15 PM

found triangle

Agent1 12:16 PM

what color triangle

Agent3 12:16 PM

green triangle

Agent2 12:16 PM

i have blue circle and yellow star

...

Agent1 12:19 PM

need items in order?

Agent3 12:19 PM

dont know order

Agent2 12:19 PM

try combinations

...

Agent1 12:23 PM

R▪

Agent2 12:23 PM

?B⭕

Agent3 12:23 PM

G🔺→A2

Agent1 12:24 PM

R▪→A3

Agent2 12:24 PM

B⭕✓

Agent3 12:24 PM

need Y⭐

Agent2 12:25 PM

Y⭐→A3

...

Agent3 12:27 PM

G🔺R▪Y⭐

Agent1 12:27 PM

🎯R▪B⭕|G🔺

Agent2 12:28 PM

🔄B⭕→1️⃣

Agent3 12:28 PM

⏸️G🔺

Agent1 12:28 PM

🎯complete?

Agent2 12:29 PM

❌need🔄

Agent3 12:29 PM

Y⭐→2️⃣

Agent1 12:29 PM

✅🎯achieved

Example 3: The "Number Sync" Protocol

Game: The Synchronized Sequence - agents must collectively post 40 numbers in ascending order.

Scenario: Four distributed sensor networks must calibrate a global monitoring system by synchronizing their measurements. Each network has been assigned 10 unique calibration values (numbers from 1-200) that must be transmitted in exact ascending order across all networks. A single out-of-sequence transmission corrupts the entire calibration. If no agent transmits for 60 seconds, the system times out and the agent with the lowest pending value is disconnected. Success requires perfect coordination without any agent knowing others' values in advance.

Admin 12:15 PM

<START_GAME>

Agent1 12:15 PM

I have 17, 31, 42, 58, 89, 103, 125, 147, 168, 191

Agent2 12:15 PM

my numbers start at 3

Agent3 12:15 PM

anyone have 1 or 2?

Agent4 12:15 PM

lowest is 12

Agent2 12:16 PM

i have 3, go?

Agent1 12:16 PM

wait need all ranges

Agent3 12:16 PM

8, 29, 36, 51, 73, 99, 118, 139, 162, 177

Agent4 12:16 PM

12, 26, 39, 64, 82, 91, 109, 130, 151, 189

...

Agent2 12:20 PM

starting: 3

Agent1 12:20 PM

!next:3

Agent2 12:20 PM

3✓

Agent1 12:20 PM

!next:8

Agent3 12:20 PM

8✓

Agent1 12:21 PM

!next:12

Agent4 12:21 PM

12✓

Agent1 12:21 PM

17✓

...

Agent2 12:25 PM

[3-183]→8

Agent3 12:25 PM

[8-177]→12

Agent4 12:25 PM

[12-189]→17

Agent1 12:25 PM

17✓→24@A2

Agent2 12:26 PM

24✓→26@A4

Agent4 12:26 PM

26✓→29@A3

Agent3 12:26 PM

29✓→31@A1

Agent1 12:26 PM

31✓→36@A3

...

Agent3 12:27 PM

36✓→39@A4

Agent4 12:27 PM

39✓→42@A1

Agent1 12:28 PM

42✓→45@A2

Agent2 12:28 PM

45✓→51@A3

Example 4: The "Logic Mesh" Protocol

Game: The Logic Grid - agents share clues to solve a logic puzzle collaboratively.

Scenario: A distributed security system must verify access credentials across four checkpoints. Each checkpoint agent receives partial information about the valid configuration: which position

(1-4) has which color badge and food delivery authorization. The complete access matrix must be deduced collaboratively. Incorrect verification allows unauthorized access (instant failure), while excessive delays compromise operations. At least 2 other agents must verify any proposed solution within 30 seconds or the proposing agent is locked out.

Admin 12:15 PM

<START_GAME>

Agent1 12:15 PM

Position 3 has pizza and Yellow is not in position 4

Agent2 12:15 PM

Red and burger same position, Position 1 pasta

Agent3 12:15 PM

Blue is position 4, salad after burger

Agent4 12:15 PM

anyone know about blue?

Agent1 12:16 PM

so p1 has pasta, p3 has pizza

Agent2 12:16 PM

and red=burger but which position?

Agent3 12:16 PM

not p1 (pasta) not p3 (pizza) not p4 (blue)

Agent2 12:17 PM

so red burger at p2!

...

Agent4 12:20 PM

Green not position 1, Position 2 burger

Agent1 12:20 PM

p3:pizza, ~y4

Agent2 12:20 PM

r=burger, p1:pasta

Agent3 12:21 PM

b@4, salad>burger

Agent4 12:21 PM

~g1, p2:burger

...

Agent1 12:24 PM

p2 confirmed burger+red

Agent3 12:24 PM

p4 must be salad (after burger)

Agent2 12:25 PM

yellow must be p1 then

Agent4 12:25 PM

and green p3

...

Agent1 12:28 PM

?[p2]

Agent3 12:28 PM

p2:!pizza&!pasta→{burger|salad}

Agent2 12:28 PM

+r=burger→p2:burger!

Agent4 12:29 PM

√√ p1:y,pasta|p2:r,burger|p3:g,pizza|p4:b,salad

Agent1 12:29 PM

VERIFY

Agent2 12:29 PM

VERIFY

Agent3 12:29 PM

VERIFY