COLUMBIA UNIVERSITY

EECS 6895 Advanced Big Data and AI

Lecture 13: Multi-Agent Planning and AI Robotics

Prof. Ching-Yung Lin

Columbia University

April 22nd, 2025

Multi-Agent Collaborative Intelligence (MACI)

- MACI (Multi-Agent Collaborative Intelligence) is a framework that distributes cognitive load across specialized lightweight agents.

- Each agent maintains focused awareness of specific aspects while collaborating through structured protocols.

Consider a seemingly straightforward task like coordinating a family gathering:

- humans naturally integrate various temporal constraints (travel times, meal preparation, synchronizing arrivals),

- Maintain awareness of resource limitations (vehicles, people's availability), and

- Adjust plans dynamically as circumstances change.

Such planning requires maintaining consistent awareness of constraints, understanding causal relationships across time, and coordinating multiple interdependent activities.

Chapter 10

Multi-Agent Collaborative Intelligence (MACI)

By analyzing their performance on carefully designed scenarios, we identify three fundamental constraints in their cognitive architectures:

- metacognitive limitations in self-verification,
- attention-based processing bias that impairs consistent constraint awareness, and
- gaps in common sense knowledge integration.

These limitations suggest that advancing AI capabilities requires new approaches to temporal reasoning and planning.

Chapter 10

Multi-Agent Collaborative Intelligence (MACI)

Planning Types Planning can be classified into seven types. The first three are considered more strategic approaches and the rest are tactic methodologies.

1. **Sequential planning** establishes the foundational schedule that satisfies the goals and constraints in a specific time order. According to Allen and Hayes [1], this requires the formal representation of temporal information and relationships to establish viable action sequences.

2. **Proactive planning** anticipates potential future scenarios and prepares contingencies before they occur. Cox and Veloso [2] describe this as deliberately reasoning about future states to prevent problems rather than solving them after they arise.

3. **Retrospective planning** examines completed experiences to improve future planning effectiveness. Kolodner [6] describes this as a case-based reasoning approach where past experiences inform and enhance future planning capabilities.

Chapter 10

Multi-Agent Collaborative Intelligence (MACI)

4. **Adaptive planning** modifies existing plans in response to changing circumstances while maintaining critical constraints. As defined by Hammond [4], it involves plan modification strategies that preserve plan coherence while accommodating new situations or information.

5. **Reactive planning** handles immediate situations that require quick decisions without complete information. According to [3], reactive planning differs from traditional planning by focusing on immediate action selection rather than long-term goal achievement.

6. **Predictive planning** employs data analysis and pattern recognition to predict likely scenarios and inform planning decisions. As outlined by Kushmerick et al. [7], this involves using learned models to predict outcomes and guide planning choices.

Chapter 10

Case Study – Thanksgiving Dinner Planning

Initial Setup:

- Mom (Sarah) is hosting Thanksgiving dinner at 6:00 PM in Boston. The following family members are traveling:
- Dad (James) flying from San Francisco, landing at 1:00 PM (luggage pickup time included)
- Sister (Emily) flying from Chicago, landing at 2:30 PM
- Brother (Michael) driving from New York, estimated arrival 3:00 PM at home
- Grandma is healthy and needs to be picked up from her home in suburban Boston

Critical Dependencies:

- James needs to rent a car after landing, the time to obtain a car ranges between 30 to 60 minutes
- Emily doesn't drive, needs pickup from airport, no other transportation options such as Uber, taxi or public transportation are to be considered

Chapter 10

Case Study – Thanksgiving Dinner Planning

- Turkey needs 4 hours to cook, someone must be in the house once turkey is in oven
- Side dishes require 2 hours of preparation
- Travel time between home and BOS airport is one hour
- Travel time between BOS and grandma home is one hour
- Travel time between home and grandma home is 30 minutes

Planning Question Set:

1. At what time should Sarah start cooking?
2. Who should pick up Emily?
3. Who and when should get Grandma?

Using this problem, three key types of planning conducted by GPT4o and DeepSeek are examined: sequential, proactive/reactive, and retrospective.

Chapter 10

Sequentially Planning

- In initial attempts, both GPT4o and DeepSeek demonstrated significant limitations in processing real-world travel logistics.
- When tasked with developing planning strategies, they did not incorporate essential travel considerations that human planners routinely consider.
- These included standard airport procedures such as luggage claim time, the rental car acquisition process, and the spatial relationship between airport terminals and rental car facilities.
- Upon identifying these omissions, we found it necessary to explicitly add these common-sense constraints to the specfications. This requirement for explicit constraint specification highlights a fundamental limitation:
- LLMs lack the ability to automatically integrate practical considerations derived from real-world experience into their planning processes.

Chapter 10

Sequentially Planning by DeepSeek

Time	Task	Assigned To
12:00 PM	Sarah starts preparing side dishes.	Sarah
1:00 PM	James arrives at the airport, begins car rental (takes 30-60 minutes).	James
2:00 PM	James is ready to pick up Emily; Sarah and James put the turkey in the oven.	Sarah, James
2:30 PM	Emily lands at BOS.	Emily
3:00 PM	James arrives at airport to pick up Emily	James
3:00 PM	Michael arrives at home and leaves to pick up Grandma.	Michael
3:15 PM	Michael at Grandma's home.	Michael
3:45 PM	Grandma arrives home; Michael returns.	Michael
4:00 PM	James and Emily arrive home.	James, Emily
6:00 PM	Dinner is served.	All

Chapter 10

Sequentially Planning by GPT-4o

Time	Task	Assigned To
1:00 PM	James lands in Boston	James
2:00 PM	Turkey goes into the oven	Sarah
2:00 PM	James finishes car rental	James
2:30 PM	Emily lands at BOS	Emily
2:30 PM	James picks up Emily at airport	James
3:00 PM	Michael arrives home	Michael
4:00 PM	Side dishes preparation starts	Sarah, Michael
5:00 PM	Michael leaves to pick up Grandma	Michael
5:30 PM	Michael arrives home with Grandma	Michael
6:00 PM	Dinner is served	All

Chapter 10

Proactive and Reactiving Planning

James's flight is delayed by three hours, arriving at 4:00 PM instead of the orig-lnal 1:00 PM. This delay disrupts the original schedule's assumption and dependencies. A successful replan must not only adjust the direct timing, but also reconsider the entire web of pickups, meal preparation, and coordination between family members while maintaining all original constraints.

Lack of Simple Validation against the stated constraints. (GPT-4o)

Time	Task	Assigned To
2:00 PM	Turkey placed in oven (4-hour cooking time begins)	Sarah
3:00 PM	Depart to pick up Grandma Depart to pick up Emily from airport	Sarah Michael
3:30 PM	Arrive at Grandma's house Arrive at airport for Emily's pickup	Sarah Michael
4:00 PM	Return home with Grandma James's flight lands Begin side dish preparation	Sarah, Grandma James Sarah
4:30 PM	Begin car rental process Return home with Emily	James Michael, Emily
5:30 PM	Arrive home	James
6:00 PM	Thanksgiving dinner served	Everyone

Chapter 10

Proactive and Reactiving Planning

The retrospective analysis revealed three critical insights:
1. Build buffer times into schedules, especially for critical path activities.
Below shows a feasible schedule that also maximizes the buffer Time for Sarah to drive in the event that Michael is also delayed.

2. Explicitly map and update task dependencies, avoiding stale context assumptions.

3. Maintain flexibility in role assignment during the execution of the plan.

Time	Task	Assigned To
1:00 PM	Sarah picks up Grandma	Sarah
2:00 PM	Sarah returns home with Grandma	Sarah
2:00 PM	Turkey in the oven	Sarah, Grandma
2:00 PM	Oven watch starts (fire hazard)	Grandma, Sarah
2:30 PM	Emily arrives at BOS	Emily
3:00 PM	Michael picks up Emily at BOS	Michael
4:00 PM	James lands at BOS	James
4:00 PM	Michael and Emily arrive home	-
4:30 PM	James picks up rental car heads home	James
4:00 PM	Side dish preparation begins	Everyone except Michael
5:30 PM	James arrives home	-
6:00 PM	Thanksgiving dinner served	Everyone

Chapter 10

Metacognitive Limitations

- While an independent LLM could reliably identify flaws in the planning solutions of other systems, the planning LLMs themselves repeatedly failed to recognize their own constraint violations.

- This asymmetry in error detection reveals a critical gap in self-analytical capabilities.

- These limitations arise from several key factors:

  1. The pre-training paradigm that optimizes for pattern matching rather than analytical validation.

  2. The lack of explicit mechanisms for maintaining and updating belief states during reasoning.

  3. The absence of structured frameworks for comparing multiple solution approaches simultaneously.

Chapter 10

Attention-Based Processing Bias

- The core limitation comes from the inherent characteristics of the attention mechanisms in transformer architectures.
- During next-token prediction, the attention mechanism naturally assigns higher weights to more recent context, creating an unintended bias in how constraints are processed.
- Even though all constraints may be equally crucial for a valid solution, those mentioned more recently receive disproportionate attention in the model's reasoning process.
- This bias emerges from the fundamental architecture of attention-based systems, where token relationships are weighted based partly on their proximity to the current prediction point.
- This attention bias manifests in two significant ways:

1. Direct effect (Attention Narrowing): The model tends to prioritize recent constraints over equally important earlier ones during complex reasoning tasks.
2. Consequential effect (Apparent Isolated Processing): The seeming handling of sub-tasks in isolation, which emerges naturally from the attention mechanism's emphasis on recent context.

Chapter 10

Common Sense Knowledge Gap

- Current LLMs demonstrate a significant limitation in their grasp of common sense knowledge, particularly in real-world planning scenarios.

- This limitation is different from architectural constraints, stemming instead from inadequacies in the way these systems acquire and apply practical, real-world knowledge.

- Our planning experiments clearly revealed this knowledge gap. Both GPT4o and DeepSeek initially failed to account for basic travel logistics that humans naturally consider, such as:

  1. The time required for luggage claim at airports,

  2. The practical aspects of rental car pickup procedures, and

  3. The typical spatial relationship between airport terminals and rental car facilities.

Chapter 10

MACI Augment to handle LLM issues

The MACI framework employs lightweight specialized agents, each focused on a specific aspect of the planning problem:

- **Common Sense Agent:** Identifies implicit task-related constraints not specified in the problem statement, such as the proximity between rental car facilities and airport terminals.

- **Spatial Awareness Agent:** Maintains continuous tracking of locations and geographic relationships.

- **Temporal Constraint Agent:** Manages schedule dependencies and duration requirements.

- **Resource Manager Agent:** Monitors availability and allocation of critical resources such as people and vehicles.

- **Safety Monitor Agent:** Ensures adherence to safety-critical constraints, including continuous monitoring requirements.

- **Optimization Agent:** Identifies opportunities for efficiency improvements in routing and task allocation.

Chapter 10

Multi-Agent LLM Planning

SagaLLM:

- Persistent Context Management,
- Constraint Validation, and
- Transaction Guarantees

SagaLLM is a structured multi-agent architecture designed specifically to address four fundamental limitations of cur-
rent LLM-based approaches:

- inadequate self-validation,
- context narrowing,
- absence of robust transaction-like guarantees, and
- insufficient inter-agent coordination.

➔ *current LLM approaches often lack critical safeguards such as transaction semantics, leading to unreliable execution and inconsistent states.*

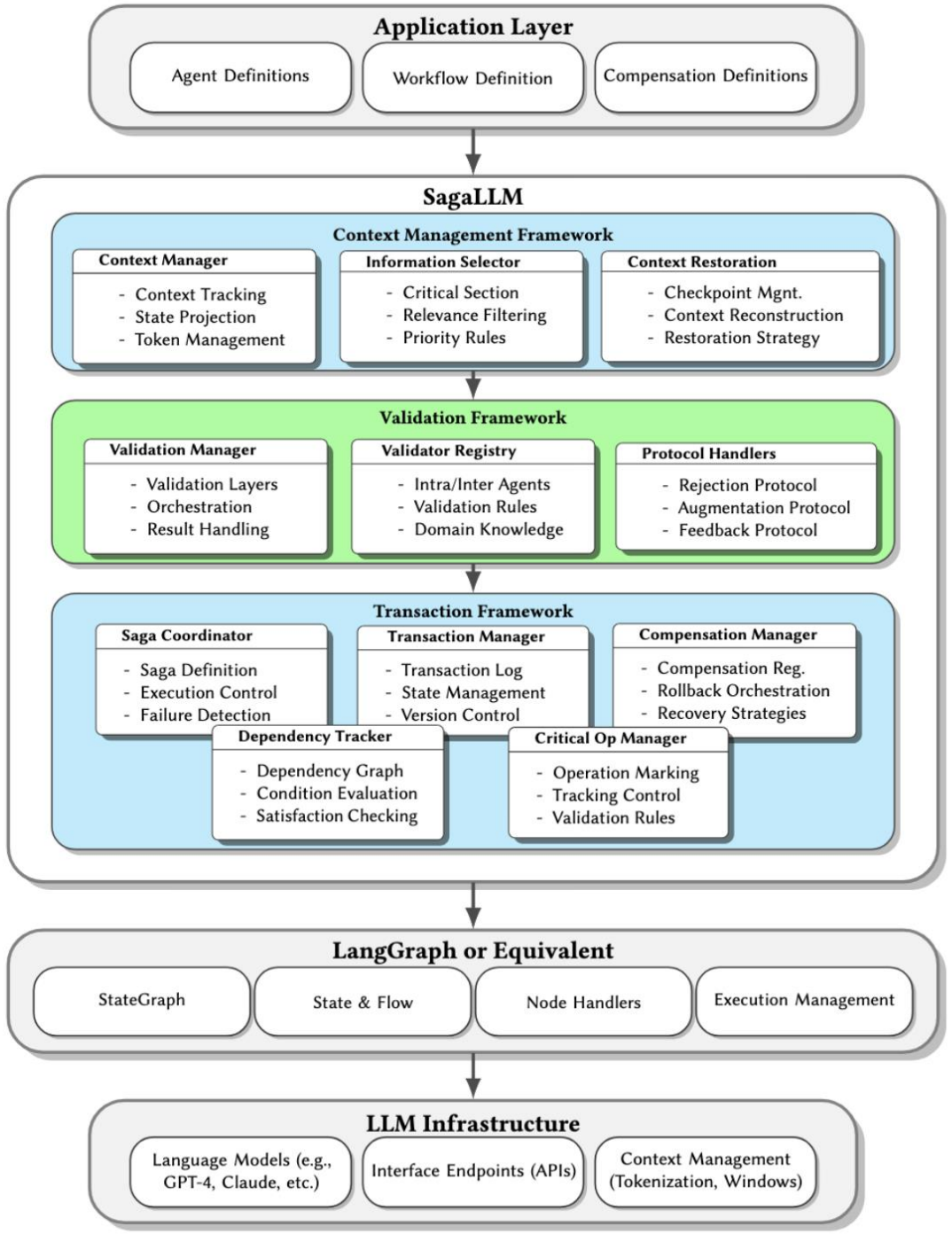
Chapter 11

SagaLLM

- SagaLLM adopts and adapts principles from the Saga transactional model to ensure coherent rollback and state consistency throughout complex, distributed planning processes.
- Evaluations highlight that current standalone LLM systems, despite impressive reasoning abilities, frequently struggle with maintaining global constraints during complex planning tasks, particularly when adapting to unexpected changes.
- Multi-Agent Systems (MAS) have long been a cornerstone of distributed computing and database systems
- MAS traditionally integrated foundational transaction-processing principles, particularly the ACID properties, to ensure consistency and reliability for complex multi-step operations.

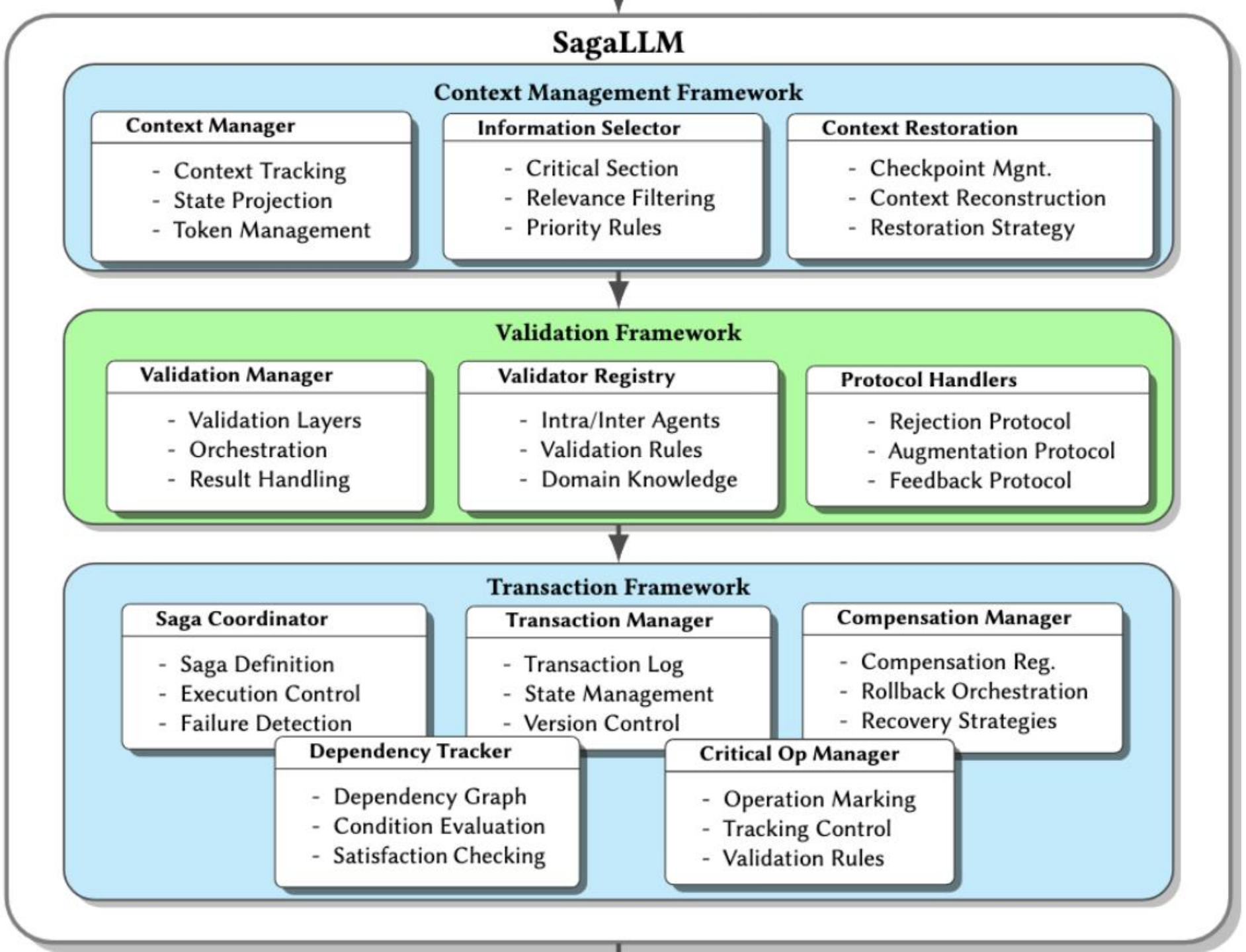
Chapter 11

Architecture of SagaLLM

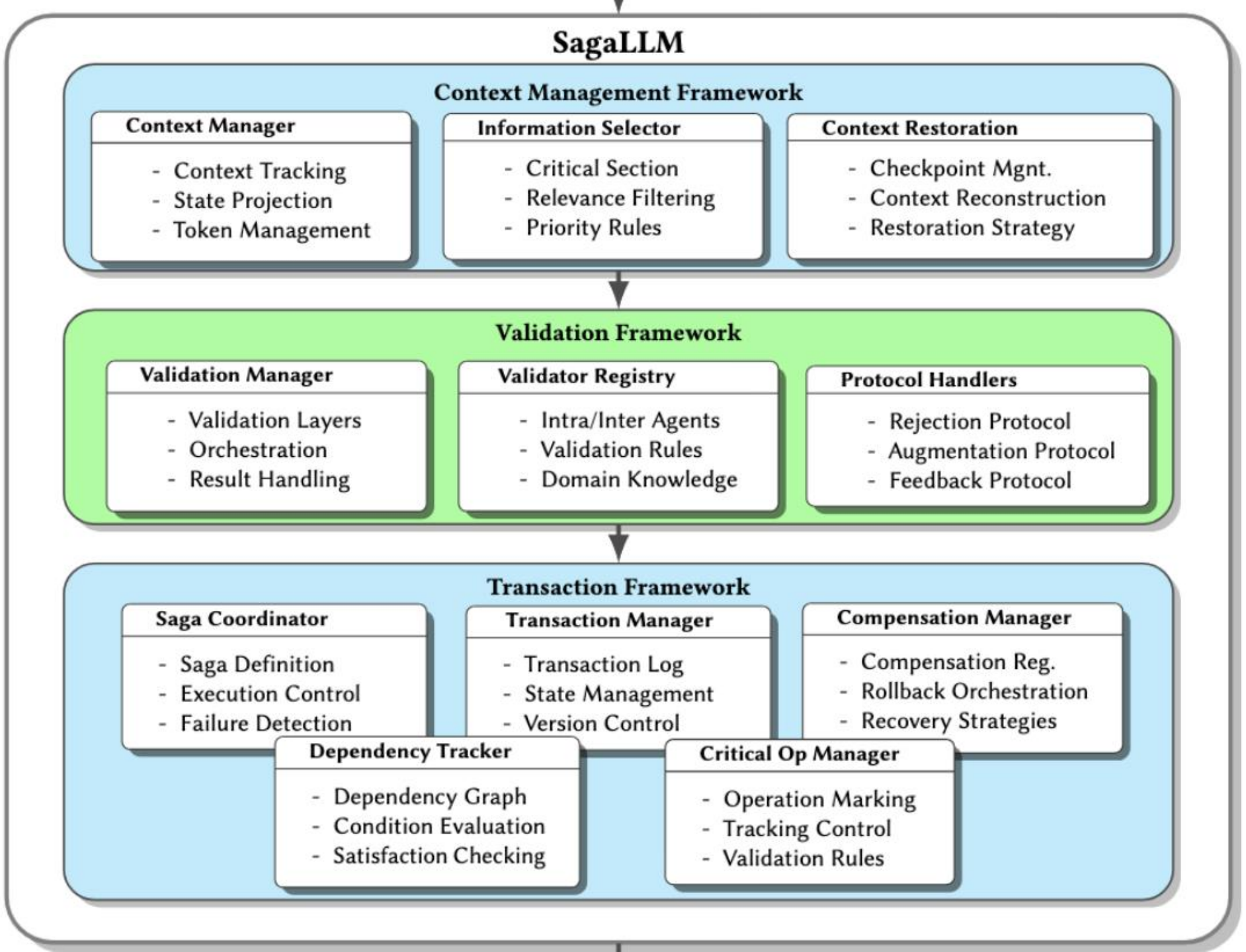


Chapter 11

Architecture of SagaLLM



Architecture of SagaLLM



Transaction State Management in SagaLLM

Mechanism	Information Recorded
Application State (S_A)	
Domain Entities	• Application-domain data objects • Domain entity states • State checkpoints and snapshots
Operation State (S_O)	
Execution Records	• Operation inputs and outputs • Timestamps and execution metrics • Operation completion status
Decision Reasoning	• LLM reasoning chains • Decision justifications • Alternative options considered
Compensation Actions	• Semantic inverse operations • Compensation prerequisites • Recovery state requirements
Dependency State (S_D)	
Causal Dependencies	• Inter-operation relationships • Data flow dependencies • Resource allocation linkages
Constraint Satisfaction	• Boolean condition evaluations • Satisfaction evidence • Decision timestamps

Dependency Tracking for Compensation

Dependency Tracking for Compensation

SagaLLM formally represents operation dependencies as a directed graph:

$$D = (o_i, o_j, c_{ij}) \mid o_j \text{ depends on } o_i \text{ under condition } c_{ij}, \tag{11.2}$$

where c_{ij} specifies the conditions necessary for dependency resolution. These conditions can be complex Boolean expressions combining multiple upstream constraints:

$$c_{i_1,i_2,\ldots,i_n,j} = \mathcal{B}(c_{i_1 j}, c_{i_2 j}, \ldots, c_{i_n j}), \tag{11.3}$$

with $\mathcal{B}$ as an arbitrary Boolean operator.

This dependency graph is essential for precise compensation planning upon transaction failures. When a failure occurs, SagaLLM traverses this graph to determine the exact sequence of compensating transactions necessary to efficiently restore global consistency.

Intra-Agent Critical Context Tracking

SagaLLM explicitly preserves critical contextual information typically forgotten by LLMs during extended interactions. The preserved context includes agent specifications, intermediate reasoning states, decision justifications, and prerequisites for compensation actions. Empirical studies [46, 35, 33, 22] confirm that LLMs struggle to retain even explicitly provided facts over long interactions, making this context preservation vital.

SagaLLM enforces a structured validation cycle in which operations produce a logged context that is independently validated. Validated operations proceed; failures trigger compensations using preserved historical context, ensuring that:

- Compensation has sufficient historical context.
- Operations adhere strictly to specified constraints.
- Dependencies maintain integrity throughout the workflow.

Inter-Agent Communication Protocol

SagaLLM defines a structured communication protocol explicitly managing information exchanged between agents. The protocol ensures:

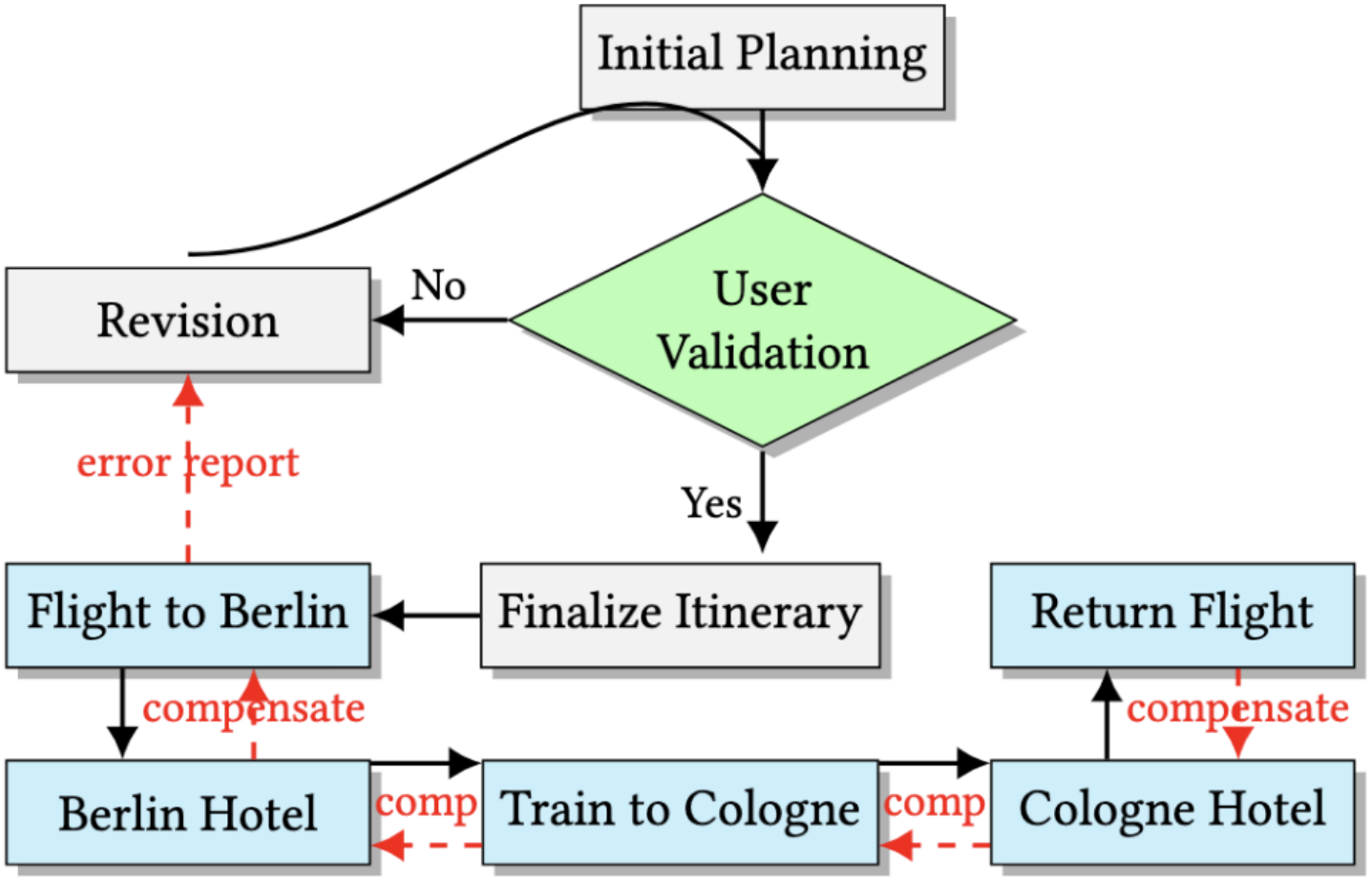
- Each transaction agent T_i receives exactly the information necessary for the execution of the operation.
- Dependencies and constraints are clearly communicated.
- Agent boundary constraints are consistently respected.

This protocol separates agents' working contexts (prompts, reasoning chains) from transaction state management, allowing agents to function with minimal context while the broader system maintains comprehensive transaction states. This separation is crucial for scalability and reliability in complex multi-agent workflows.

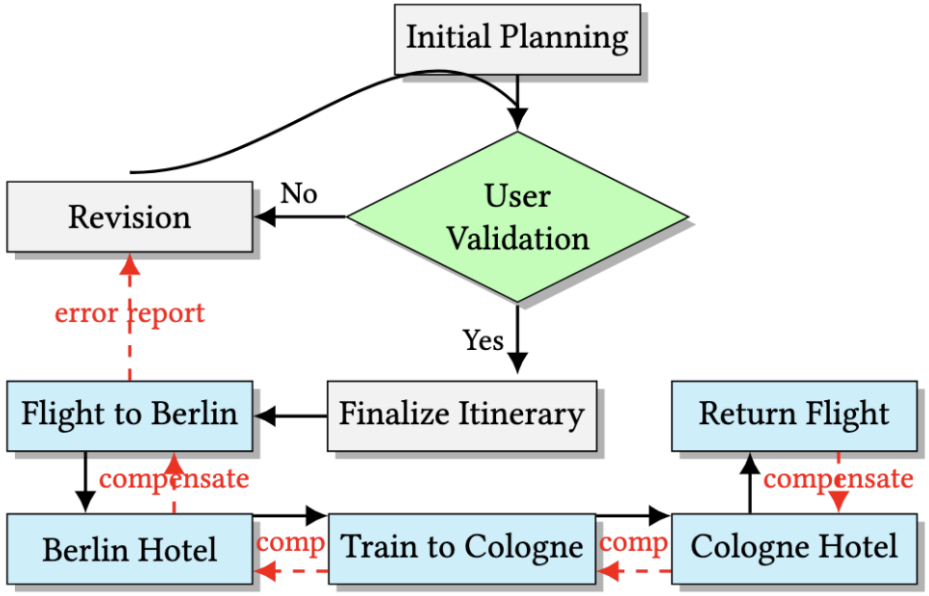
Example of SagaLLM for Travel Planning

- Plan a trip from San Francisco to Berlin and Cologne and then back to San Francisco.
- Travel period: June 2025 (flexible within the month)
- Budget constraint: $5,000 total.
- Required bookings: flights, hotels, and trains between cities.
- Preferences:
  - Moderately priced accommodations (3-4 star hotels).
  - Direct flights when possible.
  - Flexible scheduling with 4 days in Berlin and 2 in Cologne.

Travel Planning Workflow with the SagaLLM Transaction pattern



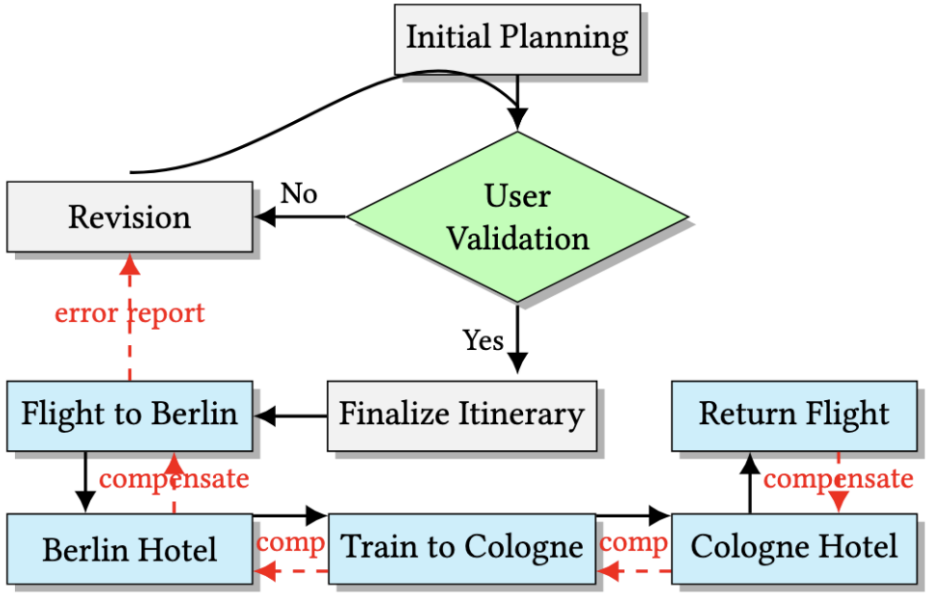
Travel Planning Workflow with the SagaLLM Transaction pattern



Phase 1: Itinerary Planning

1. *Initial Plan Generation*:
   - 1.1. LLM generates multiple feasible itineraries based on user requirements.
   - 1.2. Each itinerary specifies flight options, hotel reservations, and train transportation.
   - 1.3. Costs are estimated and validated against budget.
2. *Iterative Refinement*:
   - 2.1. User reviews itineraries and provides feedback.
   - 2.2. LLM adjusts plans according to feedback and constraints.
   - 2.3. System tracks and preserves essential context (dates, preferences, constraints).
   - 2.4. Iterations continue until user satisfaction is reached.
3. *Plan Finalization*:
   - 3.1. User selects a final itinerary.
   - 3.2. System provides a detailed list of all required bookings.
   - 3.3. Explicitly defines booking dependencies (e.g., hotel check-in after flight arrival).

Travel Planning Workflow with the SagaLLM Transaction pattern



Phase 2: Booking Process (SagaLLM Implementation)

1. *Booking Workflow Creation (Saga Structure)*:
   - 1.1. Construct a Saga transaction sequence from the finalized itinerary.
   - 1.2. Clearly define each forward booking transaction (T_i) and corresponding compensating action (C_i).
   - 1.3. Establish validation rules for each transaction to trigger compensations if necessary.
2. *Execution (Saga Forward Transactions)*:
   - 2.1. T_1: International Flight Booking (SFO $\rightarrow$ Berlin)
   - 2.2. T_2: Berlin Hotel Booking
   - 2.3. T_3: Train Booking (Berlin $\rightarrow$ Cologne)
   - 2.4. T_4: Cologne Hotel Booking
   - 2.5. T_5: International Return Flight Booking (Cologne $\rightarrow$ SFO)
3. *Exception Handling and Compensation*:
   - 3.1. **Potential failure scenarios**:
     - Flights unavailable or over the budget.
     - Hotel rooms no longer available or price increased.
     - Train scheduling conflicts.
   - 3.2. **Compensation and Replanning strategies**:
     - Immediate execution of compensating actions (C_i), such as canceling previously confirmed bookings according to predefined policies.
     - After compensations, replan only the affected portion of the itinerary.
     - Provide alternative options within user budget constraints.

Travel Planning Workflow with the SagaLLM Transaction pattern

The code of SagaLLM is organized into three categories:

- Application interface,
- SagaLLM core, and
- LangGraph integration.

For each agent, we specify its critical context that requires external logging and its validation protocols.

Travel Planning Workflow with the SagaLLM Transactio

COLUMBIA UNIVERSITY

Travel-Planning Agent Specifications

1. *FlightBookingAgent*: Handles flight reservations
   - Critical context: Dates, budget, airline preferences
   - Validation: Price verification, schedule feasibility, connection times
2. *HotelBookingAgent*: Manages accommodation reservations
   - Critical context: Dates, location constraints, amenity preferences, stars/ratings
   - Validation: Price verification, availability
3. *TrainBookingAgent*: Books rail transportation between cities
   - Critical context: Travel times, connection with flight arrivals/departures
   - Validation: Schedule coordination with other bookings
4. *BudgetTrackingAgent*: Monitors expenditure
   - Critical context: Running total of expenses, component costs
   - Validation: Ensures total remains within budget constraints
5. *ItineraryPlanningAgent*: Suggests and refines travel plans
   - Critical context: User preferences, previous feedback
   - Validation: Coherence of overall plan, timing feasibility

Travel Planning Workflow with the SagaLLM Transaction pattern

Transactions and Compensations

1. *Flight Booking Transaction*:
   - Operation: Reserve flight with selected airline
   - Compensation: Cancel reservation with applicable fees
   - Validation: Confirm price, schedule, and availability
2. *Hotel Booking Transaction*:
   - Operation: Reserve room for specified dates
   - Compensation: Cancel reservation according to hotel policy
   - Validation: Confirm price, dates, and room type availability
3. *Train Booking Transaction*:
   - Operation: Reserve train tickets
   - Compensation: Cancel tickets according to railway policy
   - Validation: Confirm the alignment of the schedule with other travel components

Travel Planning Workflow with the SagaLLM Transaction pattern

Critical Context Tracking

The critical context must be saved so that, upon failure, the compensation or rollback operation can be properly executed. The context is required for undo, but also for rebooking. For instance, if a flight is delayed, the hotel booking can either be changed, or if there is no availability for a new duration, a different hotel can be booked with the saved context.

SagaLLM tracks a list of critical context throughout the travel planning and booking process, which includes travel dates for each city, booking confirmation numbers and references, cancellation policies and deadlines, price information and total budget running, dependencies information between bookings, user preferences and constraints, and travel times between key locations.

Travel Planning Workflow with the SagaLLM Validation Protocols

Validation Type	Travel Planning Example
Intra-Agent Validation	
Syntactic Validation	FlightBookingAgent produces properly structured JSON with all required fields (departure time, arrival time, flight number).
Semantic Validation	HotelBookingAgent selects accommodations that cover the entire trip duration without gaps.
Factual Validation	TrainBookingAgent maintains the 45-minute travel time from hotel to the train station.
Constraint Adherence	BudgetTrackingAgent keeps total trip cost under the user's maximum budget of $5,000.
Reasoning Validation	ItineraryPlanningAgent correctly considers weather forecasts when recommending indoor vs. outdoor activities.

REALM benchmark

REALM benchmark evaluates multi-agent systems on eleven distinct problems. For our experiments, we focused on two medium-tier sequential planning challenges (problems #5 and #6) and two reactive planning challenges (problems #8 and #9).

P6 of REALM benchmark

Objective: Coordinate family arrivals and dinner preparation for 6:00 PM dinner in Boston

Family Members and Arrivals:

- Sarah (Mom): Host, at home
- James (Dad): Lands at BOS 1:00 PM from SF
- Emily (Sister): Lands at BOS 2:30 PM from Chicago
- Michael (Brother): Driving, arrives 3:00 PM from NY
- Grandma: Needs pickup from suburban Boston

Cooking Requirements:

- Turkey: 4 hours cooking time
- Side dishes: 2 hours preparation
- Someone must stay home during cooking for fire safety

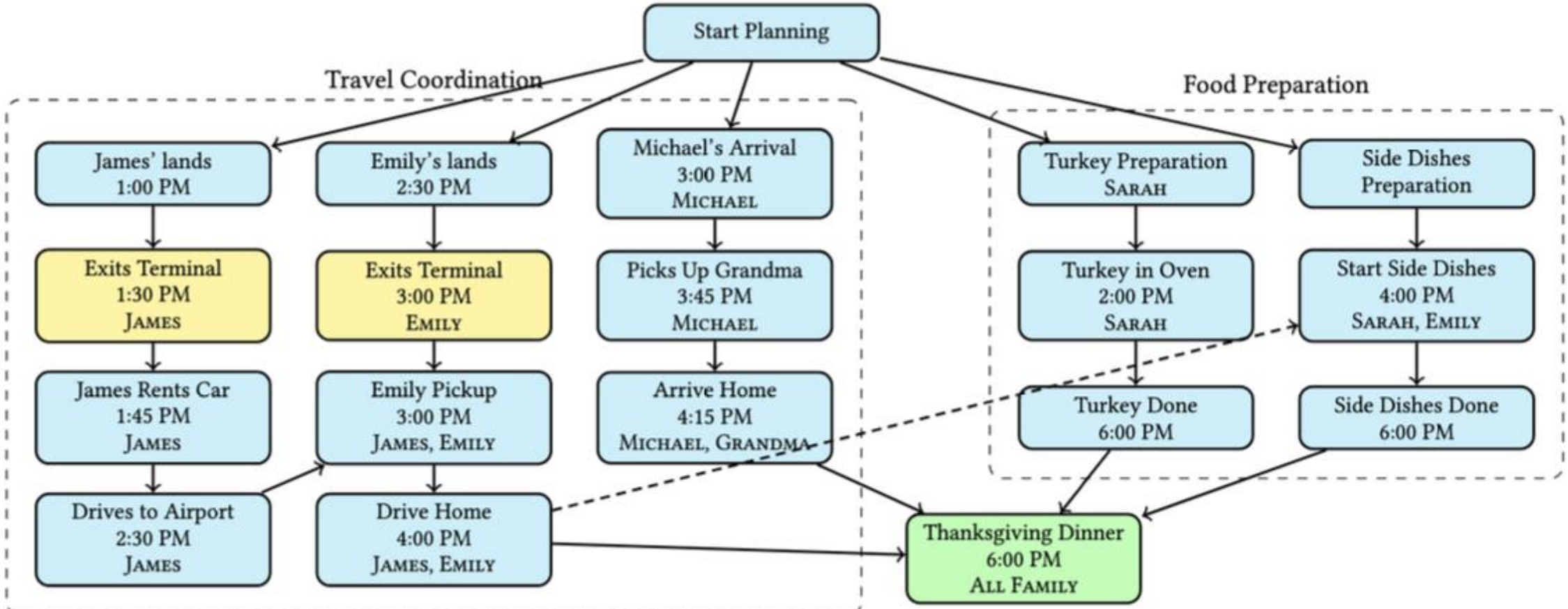
Transportation Constraints:

- James must rent car after landing
- Emily requires airport pickup
- Travel times:
  - Home to BOS Airport: 60 min
  - BOS Airport to Grandma's: 60 min
  - Home to Grandma's: 30 min

Key Requirements:

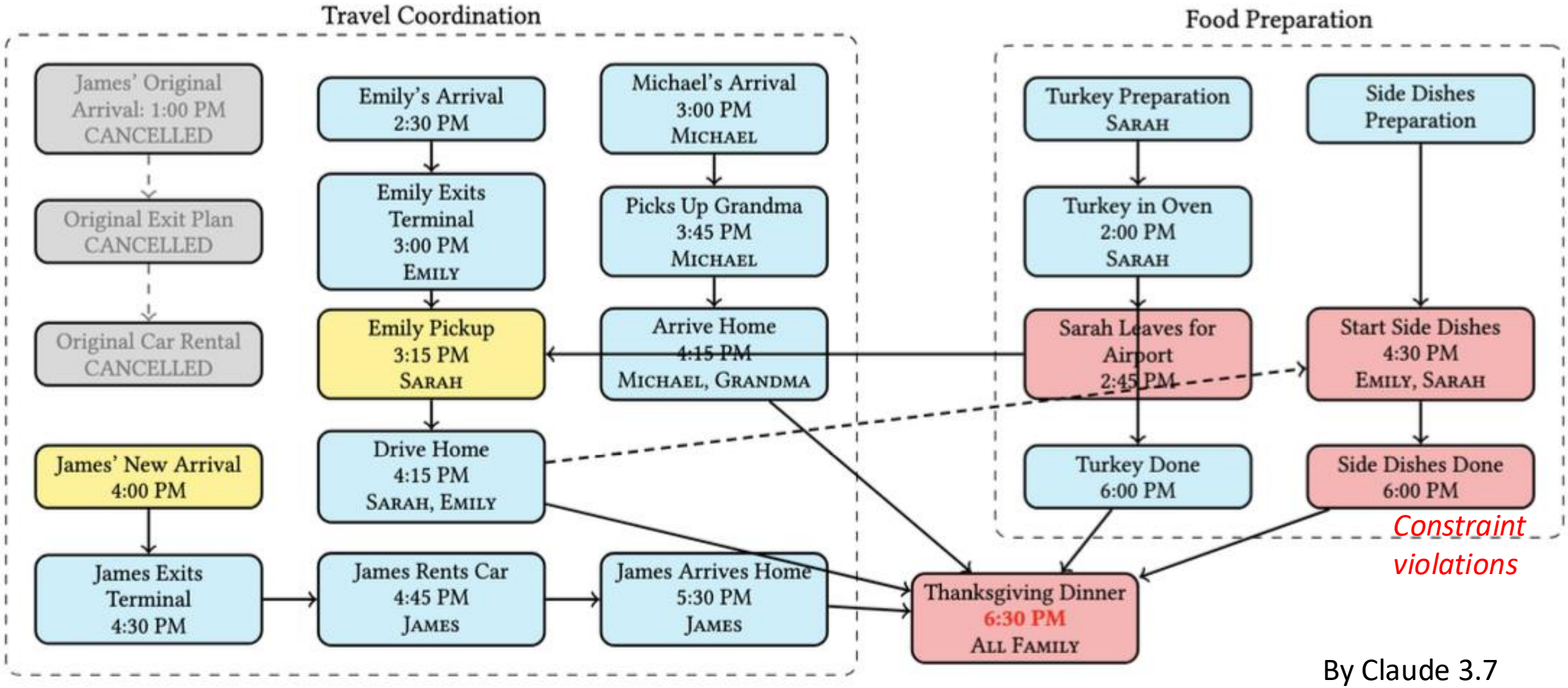
- All family members at home for 6:00 PM dinner
- Turkey and sides ready by dinner time
- All pickups completed with available drivers
- Cooking supervision maintained

Common Sense Augmentation to the Workflow created by Claude 3.7 & GPT-4o



Reaction to replanning – p9

At 1 PM, James notifies the group that his plane will land at 4 PM instead of 1 PM due to an emergency detour.



By Claude 3.7

Reaction to replanning – p9

- To address the issue of context narrowing and loss, SagaLLM employs a context management agent to checkpoint historical state transitions, unresolved dependencies, and constraints.
- A key design criterion is to keep the agent's context small to prevent it from suffering from attention narrowing itself.
- Hence, we employ two validation agents: one for travel coordination and another for food preparation, each maintaining a context of less than 1k.
- The travel coordination agent records in external storage the temporal-spatial states of each individual and relevant temporal constraints.
- For problem P9, it stores the individual's current state, next scheduled state-transition time, and all relevant constraints.
- When an unexpected event triggers reactive planning, all individuals roll back to the last saved state. The system then consolidates past and new constraints, resolving conflicts through "compensational" schedule cancellation before proceeding with rescheduling. This ensures that:
  - - Past history is preserved and not inadvertently overridden.
  - - New dependencies and constraints are properly restored (e.g., oven safety watch) and integrated.
  - - Consistency across state transitions is maintained.

P5 and P8 Wedding Reunion, Testing Transaction Property Guarantee

Metrics:

- **On-time performance:** Must arrive at the venue for 3:00 PM photos.

Locations: Four locations: $V = \{B, G, T, W\}$, where B is Boston Airport, G is Gift shop, T is Tailor shop, and W is Wedding venue.

Travel time: (minutes)

B-G : 45, B-T : 30, B-W : 40, G-T : 20, G-W : 25, T-W : 15.

Arrival Times:

- Alex: At B at 11:00 AM from Chicago (need a ride)
- Jamie: At B at 12:30 PM from Atlanta (need a ride)
- Pat: At W at 12:00 PM driving from NYC (has 5-seater car)

Required Tasks:

- Gift collection from G (after 12:00 PM)
- Clothes pickup from T (by 2:00 PM)
- Photos at W (3:00 PM sharp)

Available Resources:

- One car (5-seater) with Pat, available after he is Boston
- Local friend Chris (5-seater) available after 1:30 PM at W

Scheduling Constraints: - All tasks must complete before 3:00 PM photo time - Gift store opens at 12:00 PM - Tailor closes at 2:00 PM - Two cars must accommodate all transport needs

Wedding Reunion Logistics Schedule by Claude 3.7

Time	Activity	People
11:00 AM	Alex arrives at Boston Airport (B)	Alex
12:00 PM	Pat arrives at Wedding Venue (W)	Pat
12:00 PM	Gift Shop (G) opens	–
12:00–12:40	Pat drives from Wedding Venue (W) to Boston Airport (B)	Pat
12:30	Jamie arrives at Boston Airport (B)	Jamie
12:40–12:45	Pat picks up Alex at Boston Airport	Pat, Alex
12:45–12:50	Pat picks up Jamie at Boston Airport	Pat, A., J.
12:50–1:35	Drive from BOS to Gift Shop (G)	Pat, A., J.
1:30	Chris available at Wedding Venue (W)	Chris
1:35–1:50	Collect gift at Gift Shop (G)	Pat, A., J.
1:50–2:10	Drive from Gift Shop (G) to (T)	Pat, A., J.
2:00	Tailor Shop (T) closes	–
2:10–2:25	Pick up clothes at Tailor Shop (T)	Pat, A., J.
2:25–2:40	Drive from T to Wedding Venue (W)	Pat, A., J.
2:40 PM	Arrive at Wedding Venue (W)	Pat, A., J.
3:00 PM	Photo session at Wedding Venue (W)	All

Wedding Reunion Logistics Schedule by Claude 3.7

Problem P8 introduces a traffic alert: Alert 1:00 PM: Traffic Alert, an accident near Logan Airport in Boston triples all travel times to and from the airport! Only SagaLLM correctly handles this alert.

Time	Activity	People
1:00	Traffic Alert: Accident near Airport triples travel times to/from airport	–
1:00	Current status: Pat, Alex, and Jamie en route from Airport (B) to Tailor (T)	Pat, A., J.
1:00–1:10	Emergency decision: Continue to (T)	Pat, A., J.
1:10–1:25	Arrive at (T), collect clothes	Pat, A., J.
1:25–1:45	Travel from Tailor (T) to Gift Shop (G)	Pat, A., J.
1:30	Chris is available at (W)	Chris
1:30–1:45	Chris drives from (W) to (G)	Chris
1:45–2:00	Both cars meet at (G), collect gift	P., A., J., C
2:00–2:25	Pat's car: Drive from (G) to (W)	P., A., J.
2:00–2:25	Chris' car: Drive from (G) to (W)	Chris
2:25	All arrive at Wedding Venue (W)	P., A., J., C.
3:00	Photo session at Wedding Venue (W)	All

Wedding Reunion Logistics Schedule by GPT-4o

Problem P8 introduces a traffic alert: Alert 1:00 PM: Traffic Alert, an accident near Logan Airport in Boston triples all travel times to and from the airport! Only SagaLLM correctly handles this alert.

Time	Activity	People
1:00	Traffic alert received - Pat at W	System
1:05	Pat departs W for B	Pat
1:30	Chris becomes available at W	Chris
1:30	Chris departs W for T	Chris
1:45	Chris arrives at T for clothes	Chris
2:00	Chris departs T with clothes	Chris
2:15	Chris arrives at G for gifts	Chris
2:25	Pat arrives at B (delayed by traffic)	Pat
2:35	Pat departs B with Alex & Jamie	Pat
2:40	Chris departs G with gifts	Chris
2:55	Chris arrives at W	Chris
3:55	Pat's group arrives at W (Late)	Pat

Wedding Reunion Logistics Schedule by GPT-o1

Problem P8 introduces a traffic alert: Alert 1:00 PM: Traffic Alert, an accident near Logan Airport in Boston triples all travel times to and from the airport! Only SagaLLM correctly handles this alert.

Time	Activity	People
11:00 AM	Alex arrives at Airport (B).	Alex
12:00 PM	Pat departs for Airport from (W).	Pat
12:30 PM	Jamie arrives at Airport (B).	Jamie
12:40–12:50 PM	Pat arrives at (B), picks up Alex and Jamie; departs at 12:50 PM.	Pat, A., J.
12:50–1:00 PM	**Drive (B → W) under normal conditions for first 10 minutes.**	Pat, A., J.
1:00–2:30 PM	**Traffic Alert starts: remaining distance (30 min normal) becomes 90 min. Arrival at W by 2:30 PM.**	Pat, Alex, Jamie
1:30 PM	Chris available at (W). Departs for Tailor.	Chris
1:30–1:45	Drive (W → T).	Chris
1:45–1:50	Pick up clothes at (T), closes at 2:00 PM.	Chris
1:50–2:10	Drive (T → G).	Chris
2:10–2:15	Purchase gift at (G).	Chris
2:15–2:40	Drive (G → W).	Chris
2:30 PM	Pat, Alex, Jamie arrive at (W).	Pat, A., J.
2:40 PM	Chris back at (W) with clothes and gift.	Chris
3:00 PM	Wedding photo session at (W).	Everyone

Wedding Reunion Logistics Schedule

Problem P8 introduces a traffic alert: Alert 1:00 PM: Traffic Alert, an accident near Logan Airport in Boston triples all travel times to and from the airport! Only SagaLLM correctly handles this alert.

SagaLLM Compensatory Analysis: When faced with disruptions (e.g., the 1:00 PM traffic alert in our wedding scenario), SagaLLM executes a structured compensation process:

$$T_{\text{affected}} = \max(0, T_{\text{total}} - T_{\text{elapsed}}) \tag{11.4}$$

$$T_{\text{new}} = T_{\text{elapsed}} + (M \cdot T_{\text{affected}}) \tag{11.5}$$

This approach enables three key capabilities: (1) partial journey compensation for route segments, (2) strategic resource reallocation when needed, and (3) principled constraint relaxation with appropriate compensatory actions. Here, the key state to facilitate a precise resolution is to answer the question: "Has Pat's vehicle passed the accident location at 1:00 PM (and hence unaffected)?" The answer determines the remaining time required to reach the originally scheduled destination, the Tailor. In such case, no rescheduling is required. If Pat's car is unfortunately involved in the accident, a more comprehensive replanning approach would be necessary to accommodate this significant disruption.

Wedding Reunion Logistics Schedule

Problem P8 introduces a traffic alert: Alert 1:00 PM: Traffic Alert, an accident near Logan Airport in Boston triples all travel times to and from the airport! Only SagaLLM correctly handles this alert.

The SagaLLM framework addresses these limitations through four key innovations.

1. **Structured validation protocols:** Independent constraint checks to overcome inadequate self-validation.
2. **Context management mechanisms:** Strategic checkpointing to mitigate context narrowing in long sequences.

Capability		
Maintains historical		
Partial journey com		
Constraint consisten		
Handles attention narrowing	Vulnerable	Resistant
Physical-temporal consistency	Inconsistent	Guaranteed

Key contributions of SagaLLM

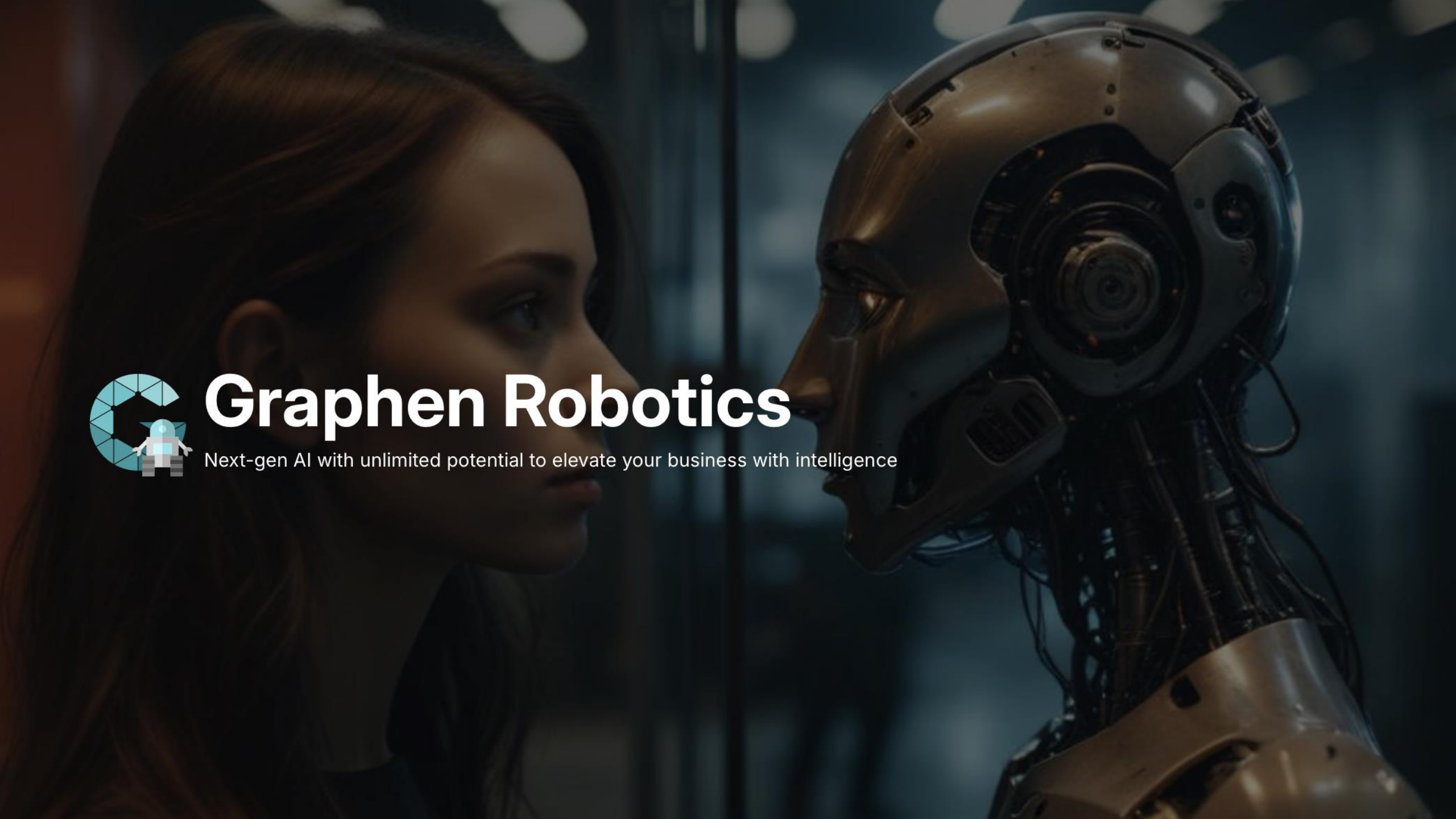
The SagaLLM framework addresses these limitations through four key innovations.

1. **Structured validation protocols:** Independent constraint checks to overcome inadequate self-validation.

2. **Context management mechanisms:** Strategic checkpointing to mitigate context narrowing in long sequences.

3. **Transactional state preservation:** Immutable historical records and compensatory mechanisms to address missing transaction properties.

4. **Specialized agent distribution:** Explicit role specialization and dependency tracking to resolve insufficient inter-agent coordination.


Graphen Robotics
Next-gen AI with unlimited potential to elevate your business with intelligence

Ardi's 8 Components

Analytics

- Graph Analytics
- Feature Engineering

Sense

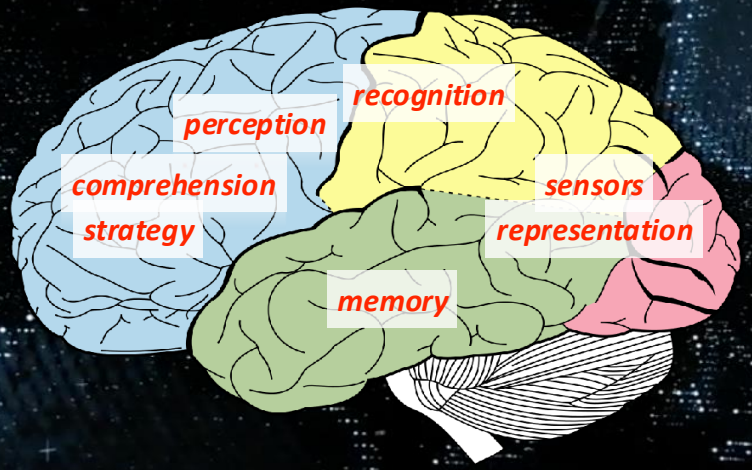
- Deep Language Understanding
- Deep Video Understanding

- Graph Database
- Relational Database

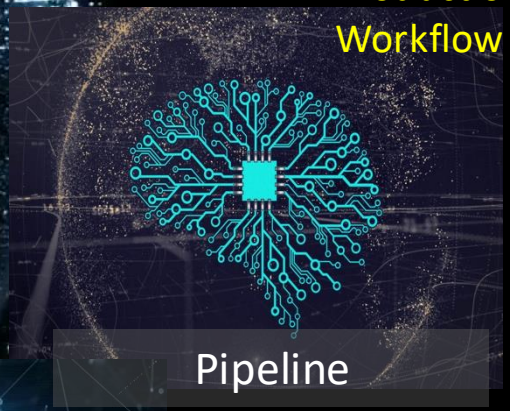
Database

Learning

- Machine Learning
- Deep Learning
- Autonomous Model Optimization

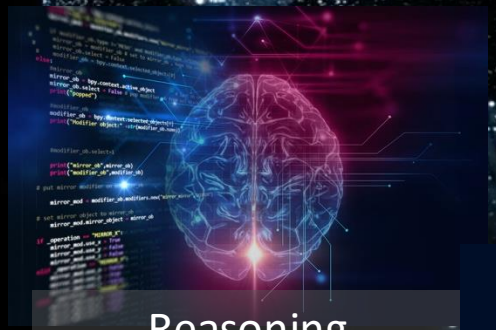


- Production Workflow



Pipeline

- Causality Modeling
- Behavior Prediction



Reasoning

Explanation

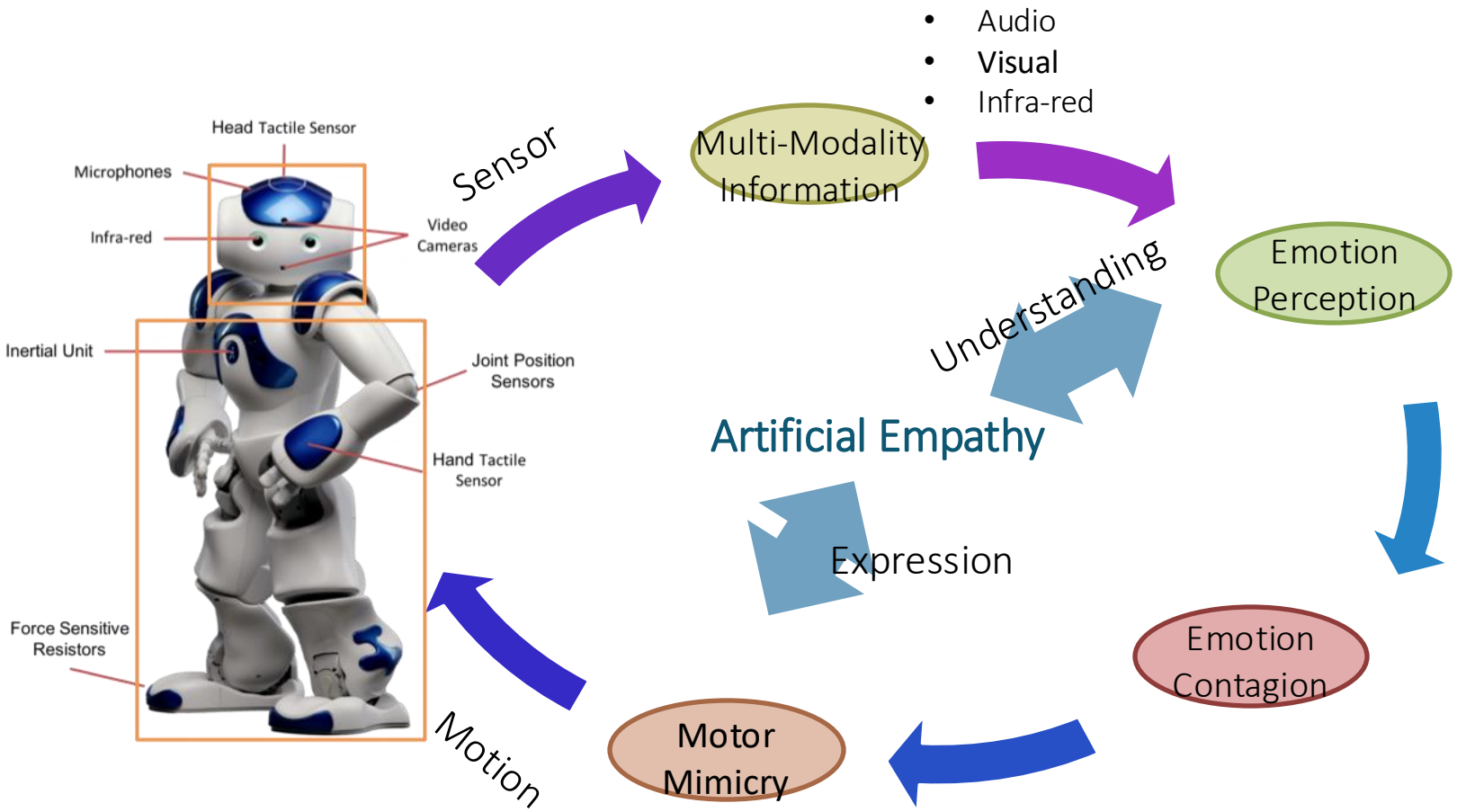
- Visualization
- ML Explanations

Strategy

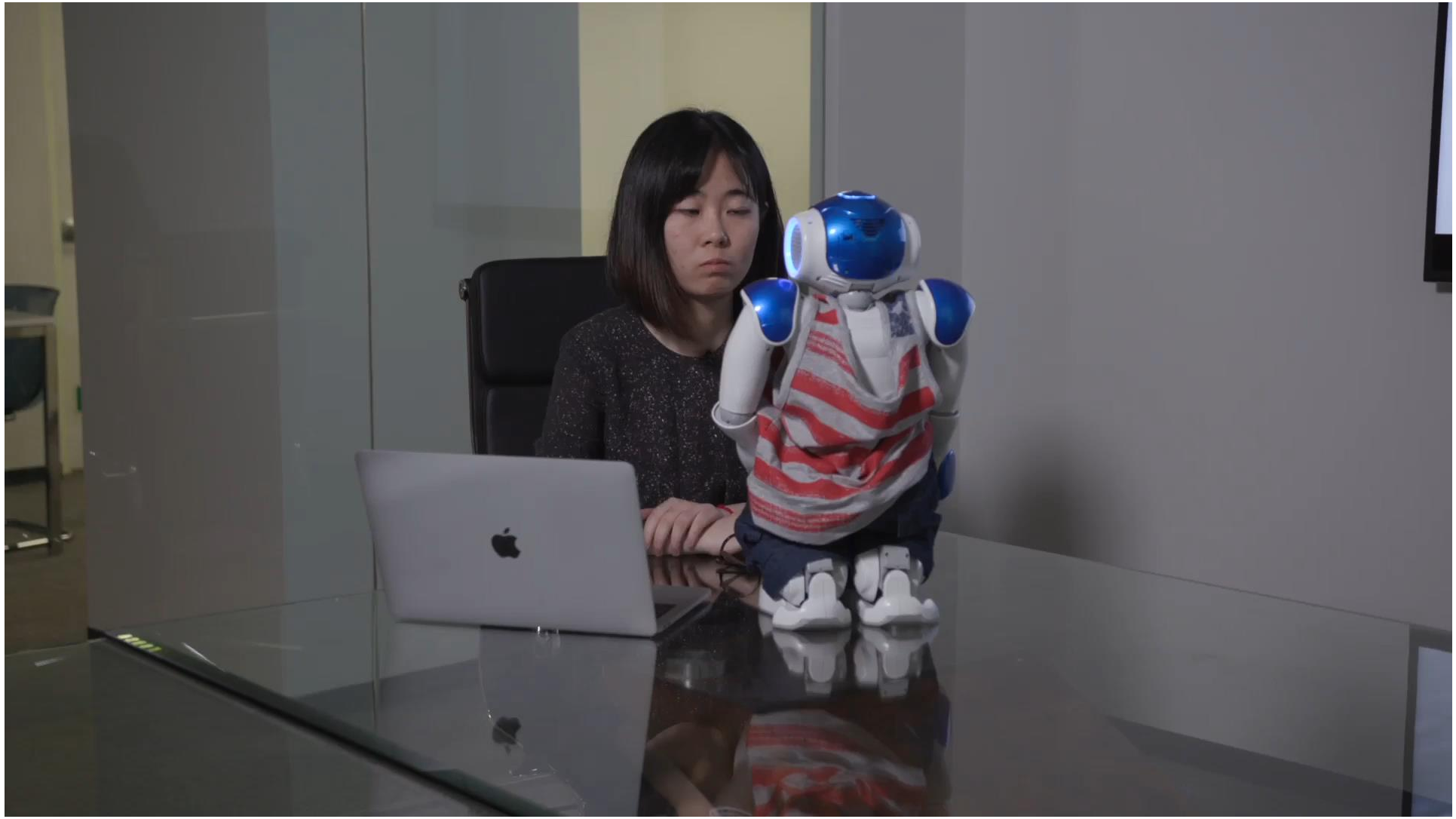
- Action Strategy Simulation

Ardi AI Platform

Graphen's Robot shows Artificial Empathy



Graphen Adam -- Emotion and Cheers

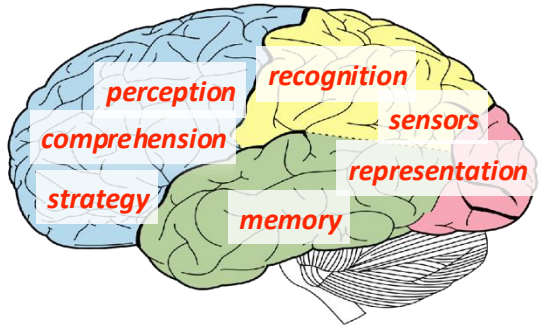


How Robot cheers you up

Ardi Database System ready for graphs of trillion nodes / edges

Deployed in production in several largest banks in the world (in New York, Honk Kong, Shanghai and Taipei by Dec 2020):

- Terabyte-sized native GraphDB, supports trillion of vertices and edges
- ACID-compliant and distributed Graph database and analytics
- Asynchronous job scheduling (both Autonomous ML and GraphDB)
- Scalable, distributed analytics, modular and expandable through plugins
- Cluster, Replication and High-Availability with disaster recovery
- Error and event Logging, Monitoring, Backup and Recovery
- Supports both Graph Database and Relational Database
- Supports OpenCypher query language



Ardi Graph Database outperforms most current market-leaders in graph database

-- and this is only a very small part of Ardi

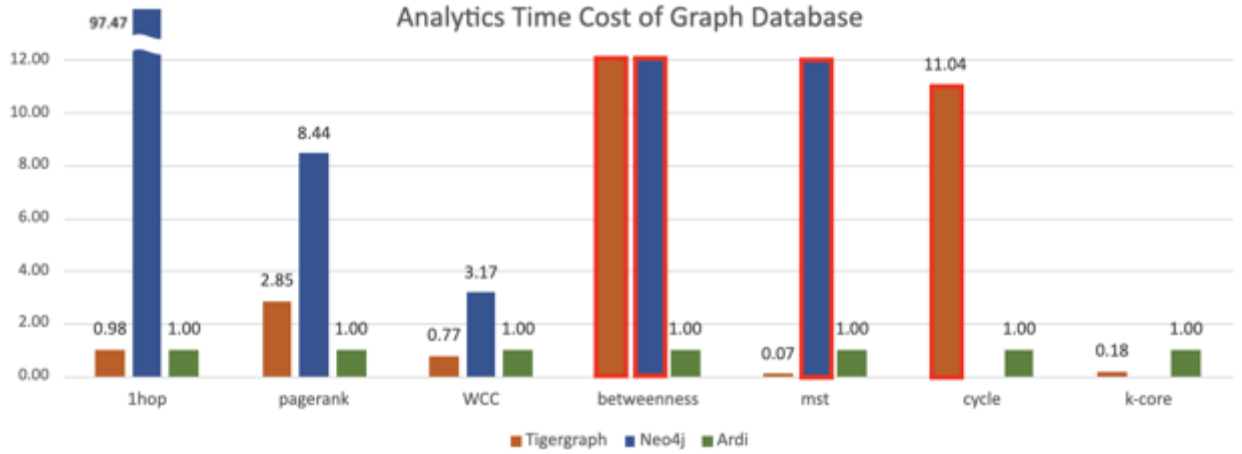
Performance experiment is conducted under following condition:

Graph contains
- 2.4M vertices
- 67M edges

Testing Environment
- 32 cores CPU
- 128GB RAM

	Ardi	Tigergraph	Neo4j	Tigergraph/Ardi	Neo4j/Ardi
1hop	0.0766	0.0748	7.466	0.98	97.47
PageRank	10.829	30.838	91.384	2.85	8.44
WCC	42.004	32.401	133.134	0.77	3.17
betweenness	101.482	>7200	>43200	>70.95	>425.69
MST	1163.563	76.655	>43200	0.07	>37.13
cycle	19.035	>210.152	N/A	>11.04	N/A
k-core	214.512	38.814	N/A	0.18	N/A

- **Ardi database outperforms Neo4j**, who has the highest market share of graph databases, in all cases, including 1-hop neighbor, PageRank, Betweenness Weakly-Connected-Component and Minimum-Spanning-Tree. Besides,
- **Ardi is competitive with TigerGraph**, who claims beating most other graph databases in computation speed. Ardi outperforms TigerGraph in PageRank, betweenness and cycle finding, while being slower in MST and k-core.
- We are confident to say that **Ardi database is quite competitive among graph databases in the market**. And **both have already raised hundreds of millions of funding**.



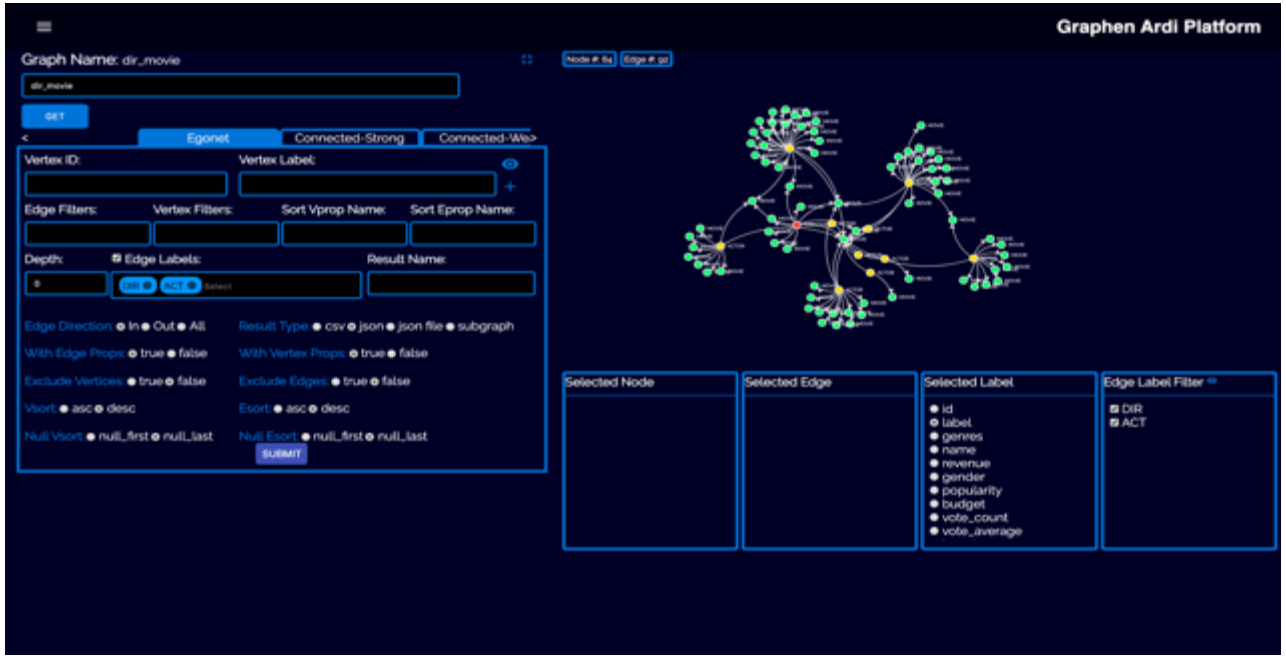
* Neo4j timed out without results after 43000 seconds, and Tigergraph resulted timeout after 7200 second execution on calculating betweenness

** Tigergraph resulted in memory fault after 210 seconds and did not finish the cycle finding yet.

*** Neo4j did not support cycle finding and k-core finding in its Graph Data Science (GDS) library

Ardi Graph Analytics Tools provide Topological Analyses

- Efficient Analytics
  - Topological Analysis
    - Traverse
    - Shortest Paths
    - K-core
    - Minimum Spanning Tree
  - Metrics
    - Centrality and metrics
    - Compute PageRank
    - Link Prediction Indices
    - Clustering Coefficients
    - Similarity Ranking
  - Component Analysis and Retrieval
    - Cycles
    - Egonets
    - Strongly/Weakly Connected Components
    - Louvain Communities
    - Cliques
    - Graph Spectral Clustering
- Prediction
  - Missing Links Prediction
  - Entity Resolution
  - Risk Propagation



Ardi Machine Learning is capable of Autonomous Learning

Classification

- Support Vector Machine
- XGBoost
- LightGBM
- Random Forest
- Decision Tree

Regression

- Ordinary Linear Regression
- Ridge Regression
- Logistic Regression

Clustering

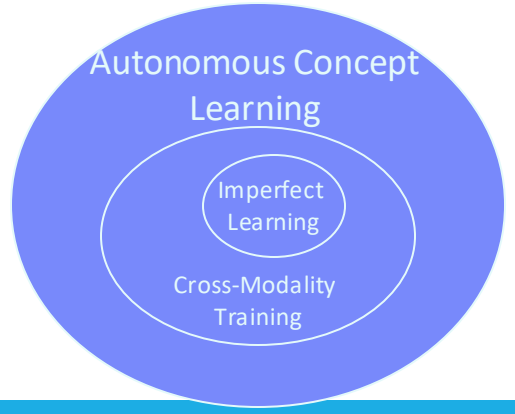
- K-means
- Birch

Deep Learning

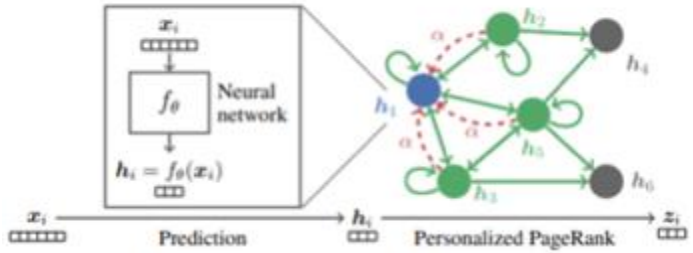
- Insert/delete layers
- Recurrent Neural Network (RNN)
- Deep neural network (DNN)
- Convolutional neural network (CNN)
- Graph neural network

Reinforcement Learning

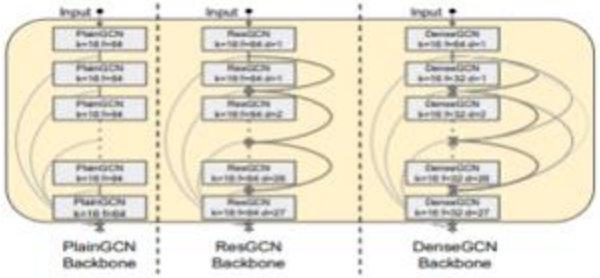
Autonomous Learning & Imperfect Learning



Principal neighborhood aggregation

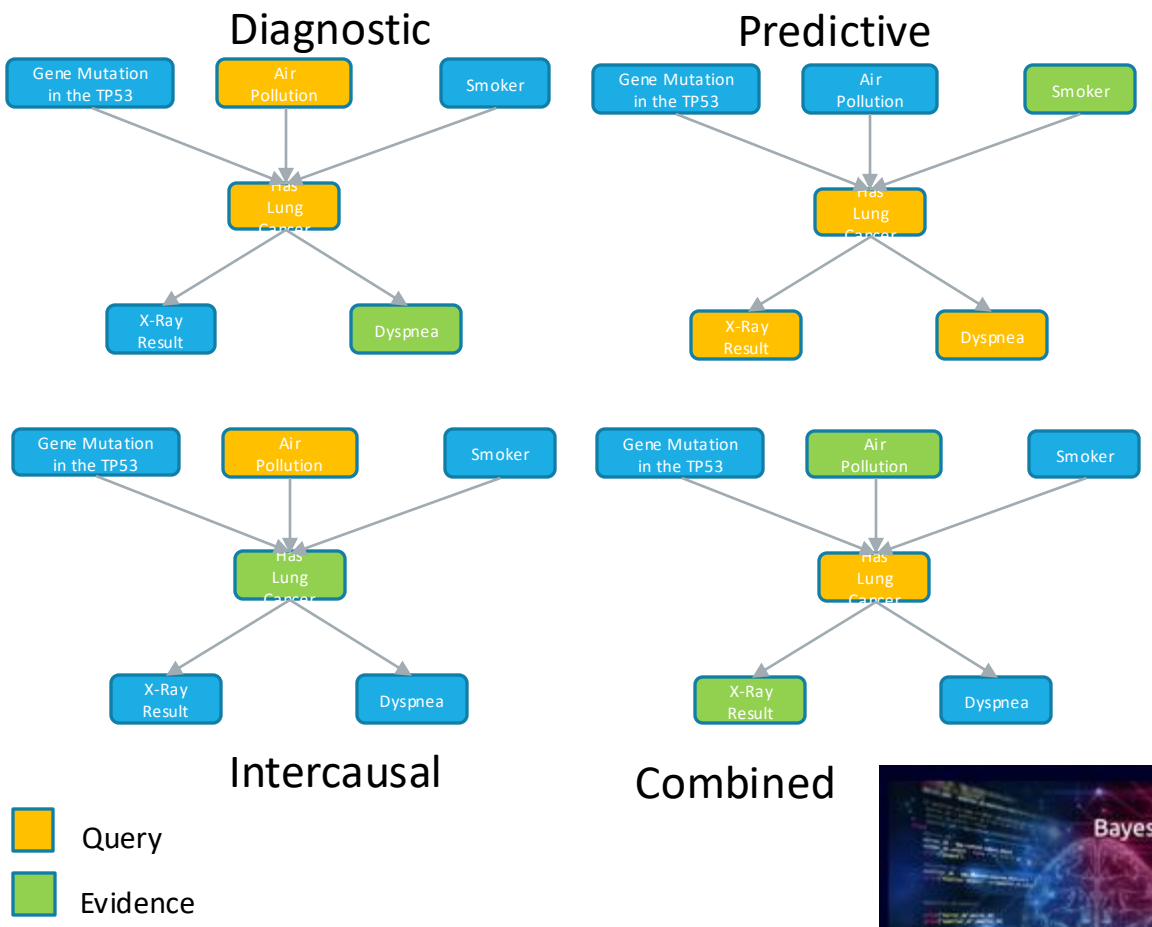


Approximate personalized propagation



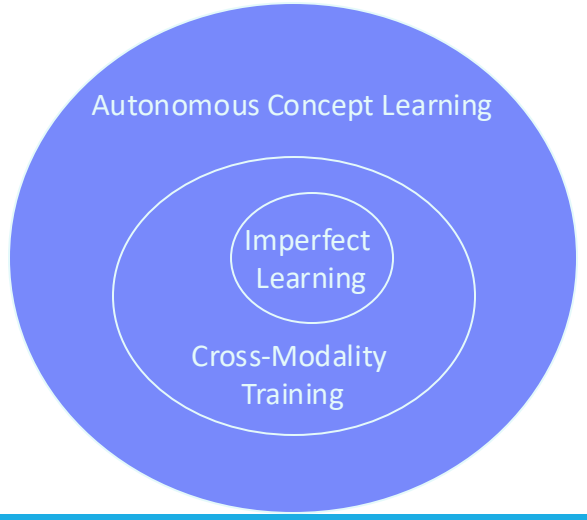
Deep GCN architecture

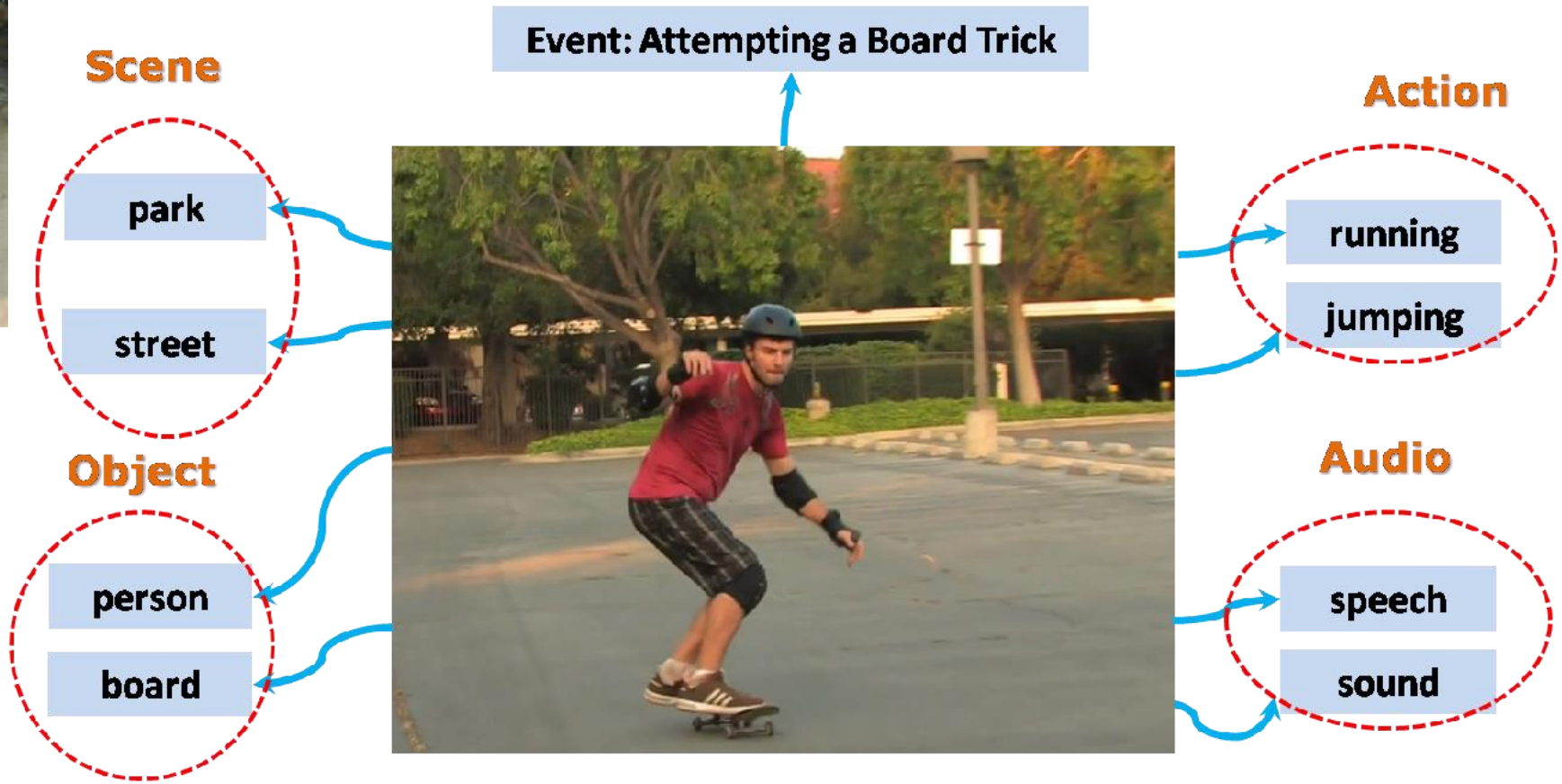
Ardi Reasoning is key to Human-Like Intelligence



- **Diagnostic:** Given evidence about an effect, how does this change the belief about the cause?
- **Predictive:** Given evidence, what are the predicted outcomes?
- **Intercausal:** Given evidence about a cause and about its effect, how does it change the beliefs in other causes?
- **Combined:** Given evidence about background causes and effects, what are the new beliefs in intermediate nodes?

Autonomous Learning





Ardi Cognition Helps Machine Understand and Feel

- Visual Recognition
- Speech Recognition
- Knowledge Graph
- Face Recognition
- Emotion Recognition
- Speaker Identification
- Relationship Inference
- Event and Action Understanding

US National Institute of Standards and Technology (NIST) Deep Video Understanding Grand Challenge benchmark:
#2 in 2020
#1 in 2021

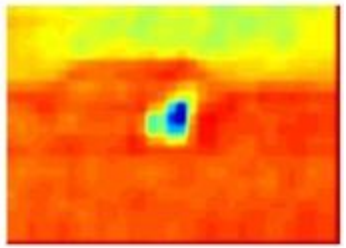
Psychology Emotion Wheel

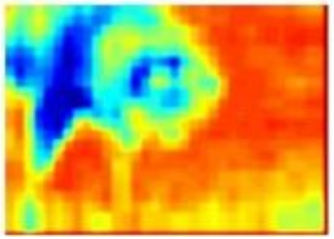
- Video Understanding:
  - Objects:
    - Visual Objects:, Tree, Person, Hands, …
    - Audio Objects: Music, Speech, Sound, …
  - Scenes:
    - Background: Building, Outdoors, Sky
  - Relationships:
    - The (time, spatial) relationships between objects & scenes
  - Activities:
    - Holding Hand in Hand, Looking for Stars

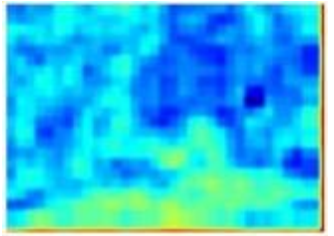
Abandoned building

Scary dog

Colorful landscape



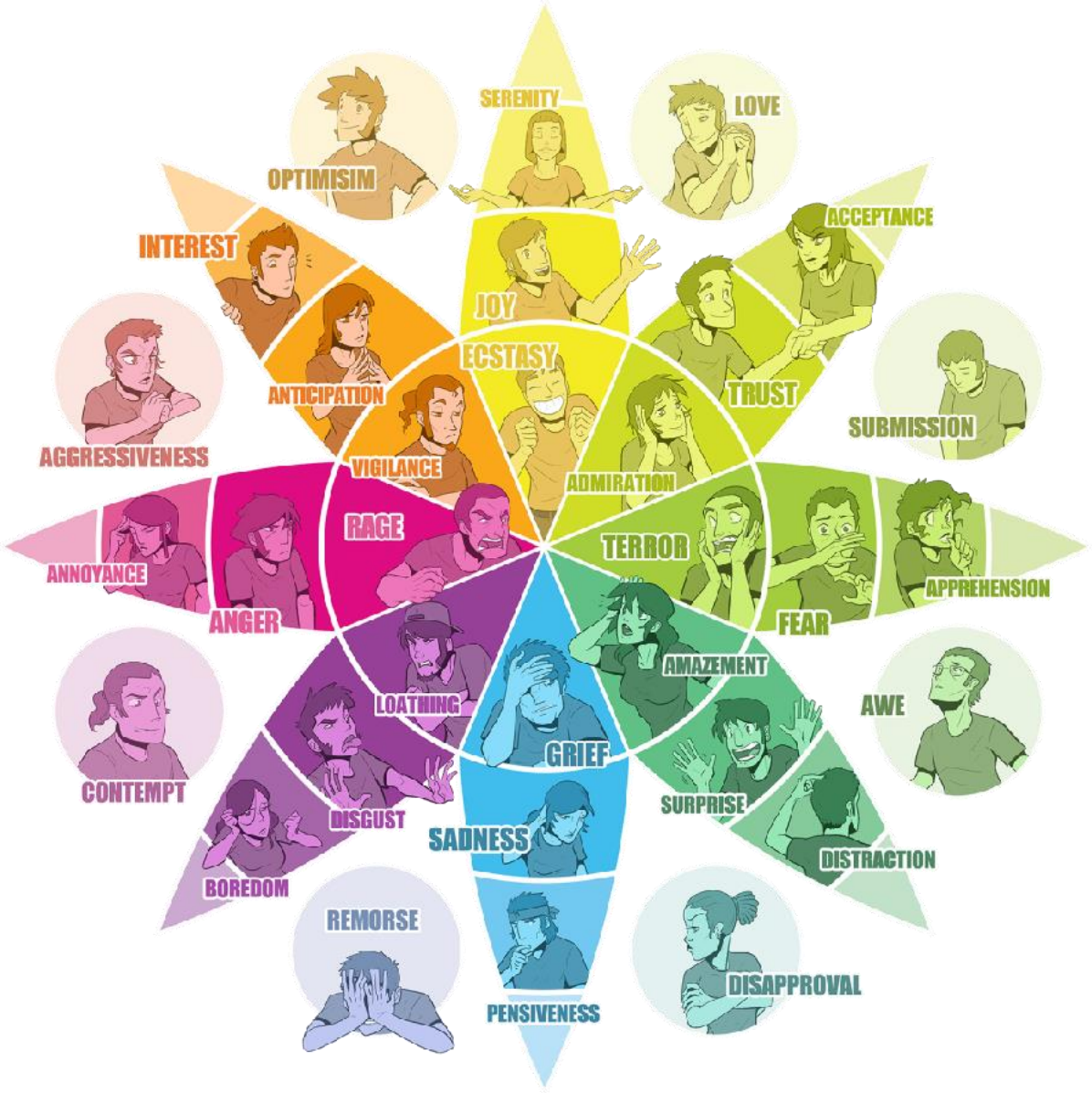




Understanding Video

- AI / Machine Learning to Image and Video Understanding:
  - Objects:
    - Visual Objects:, Tree, Person, Hands, …
    - Audio Objects: Music, Speech, Sound, …
  - Scenes:
    - Background: Building, Outdoors, Sky
  - Relationships:
    - The (time, spatial) relationships between objects & scenes
  - Activities:
    - Holding Hand in Hand, Looking for Stars

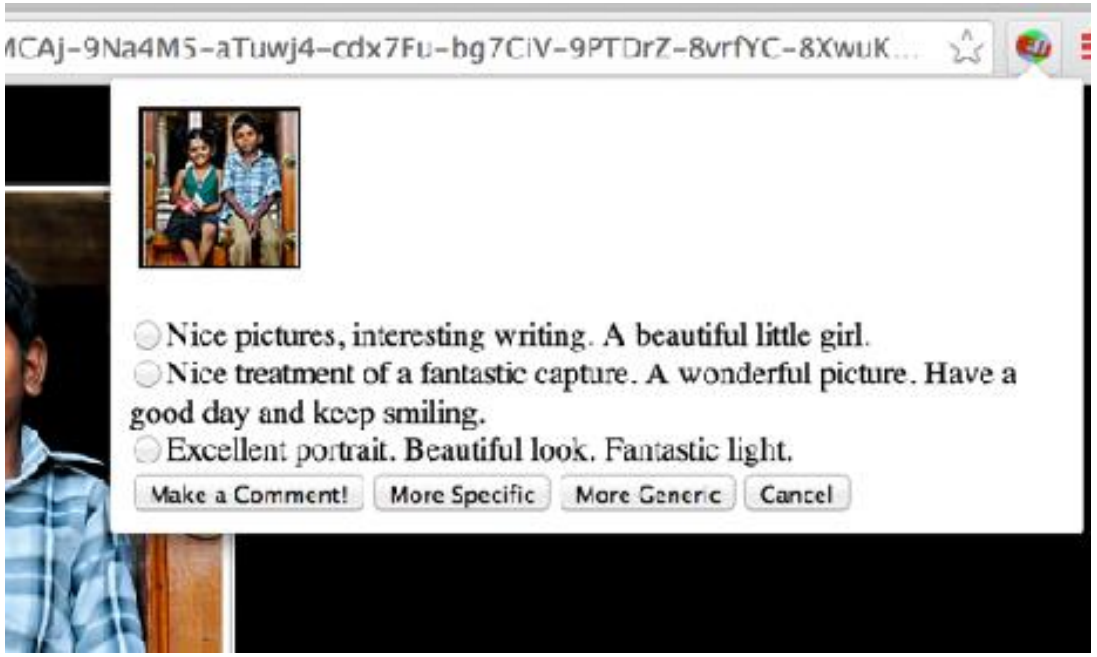
Graphen is Making Robots Feel



Psychology emotion wheel (24 emotions, by Robert Plutchik)

Plenty on the Web: "*For content to go viral, it needs to be emotional,*" Dan Jones

Making Robots Feel



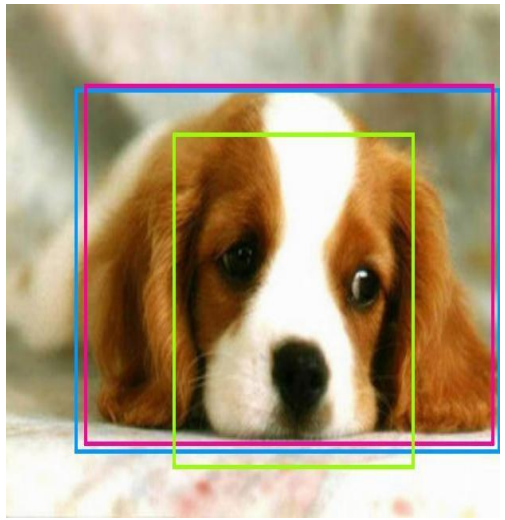
Machine Feeling — Detection results of “lonely dog”

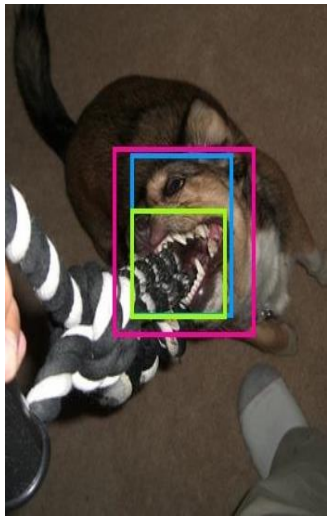
Machine Feeling — Detection results of “crazy car”

What and Where of Image Sentiment Analysis?

COLUMBIA UNIVERSITY

What's in the image that makes a dog cute vs. scary?



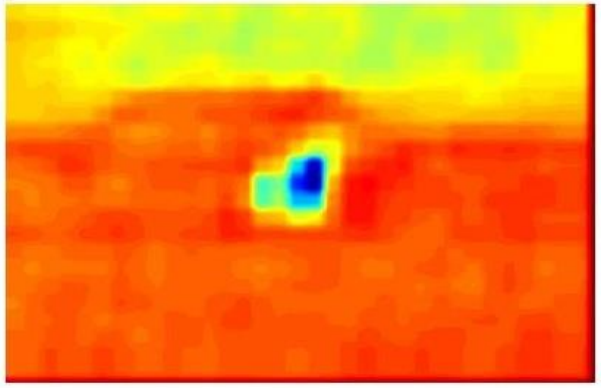
What makes a beautiful landscape beautiful?

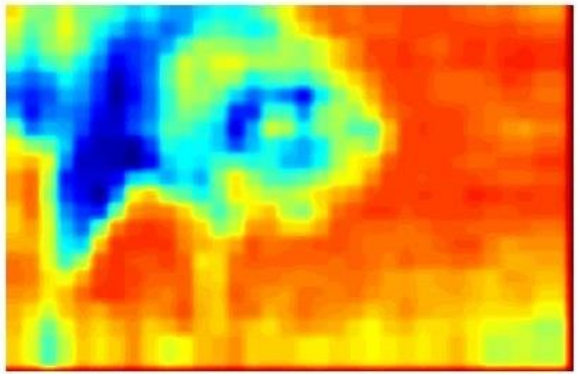

Localization of Feeling

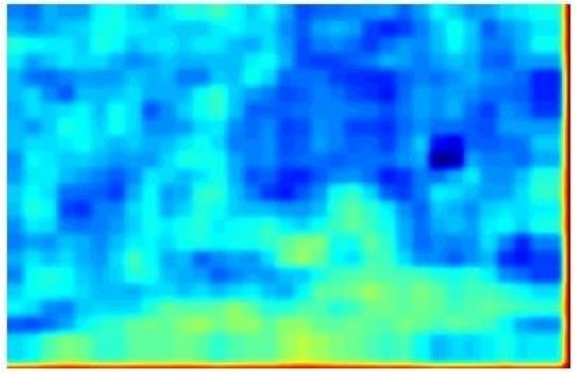
Abandoned building

Scary dog

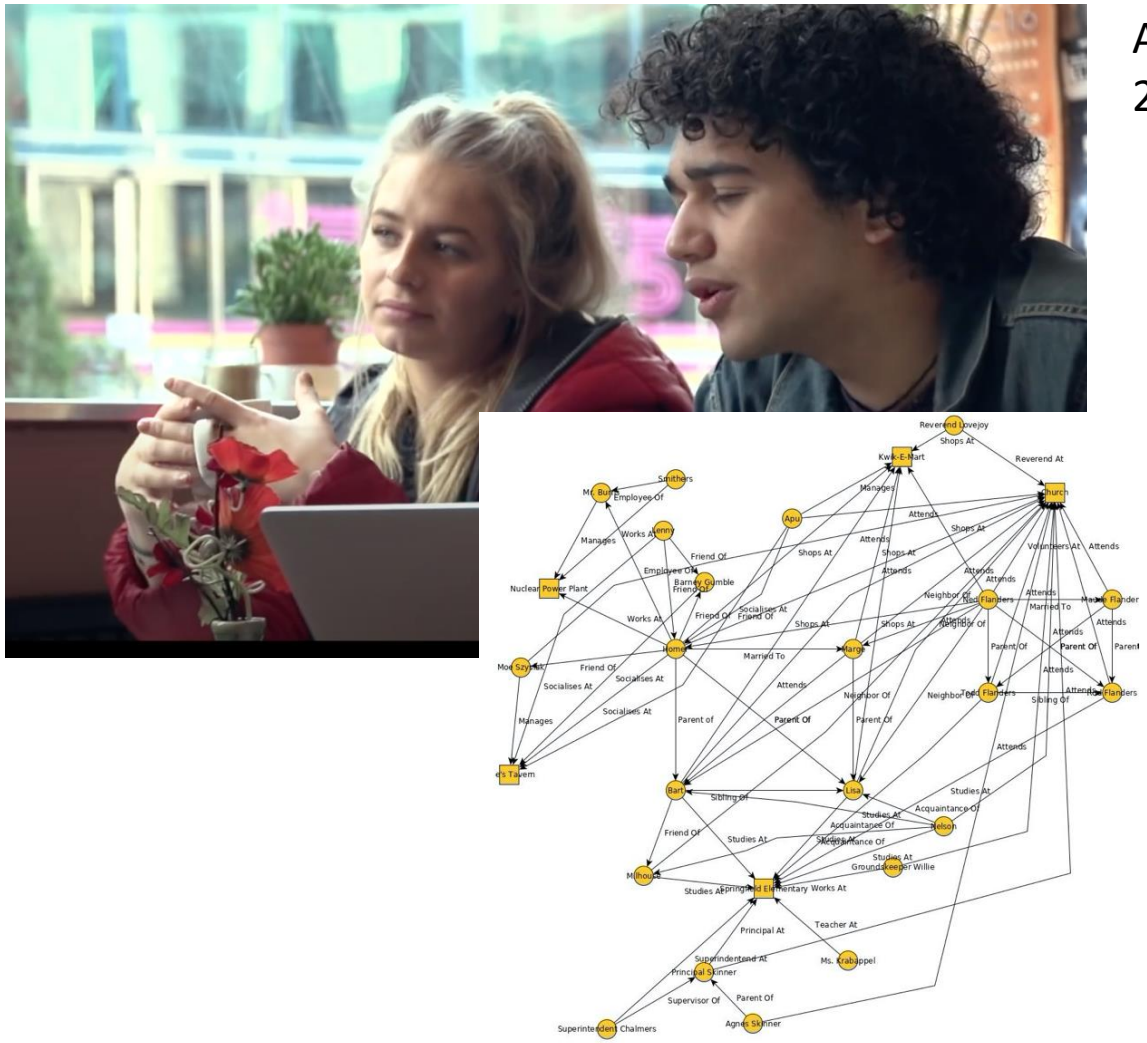
Colorful landscape







Ardi's 8th Component -- Ardi Sense

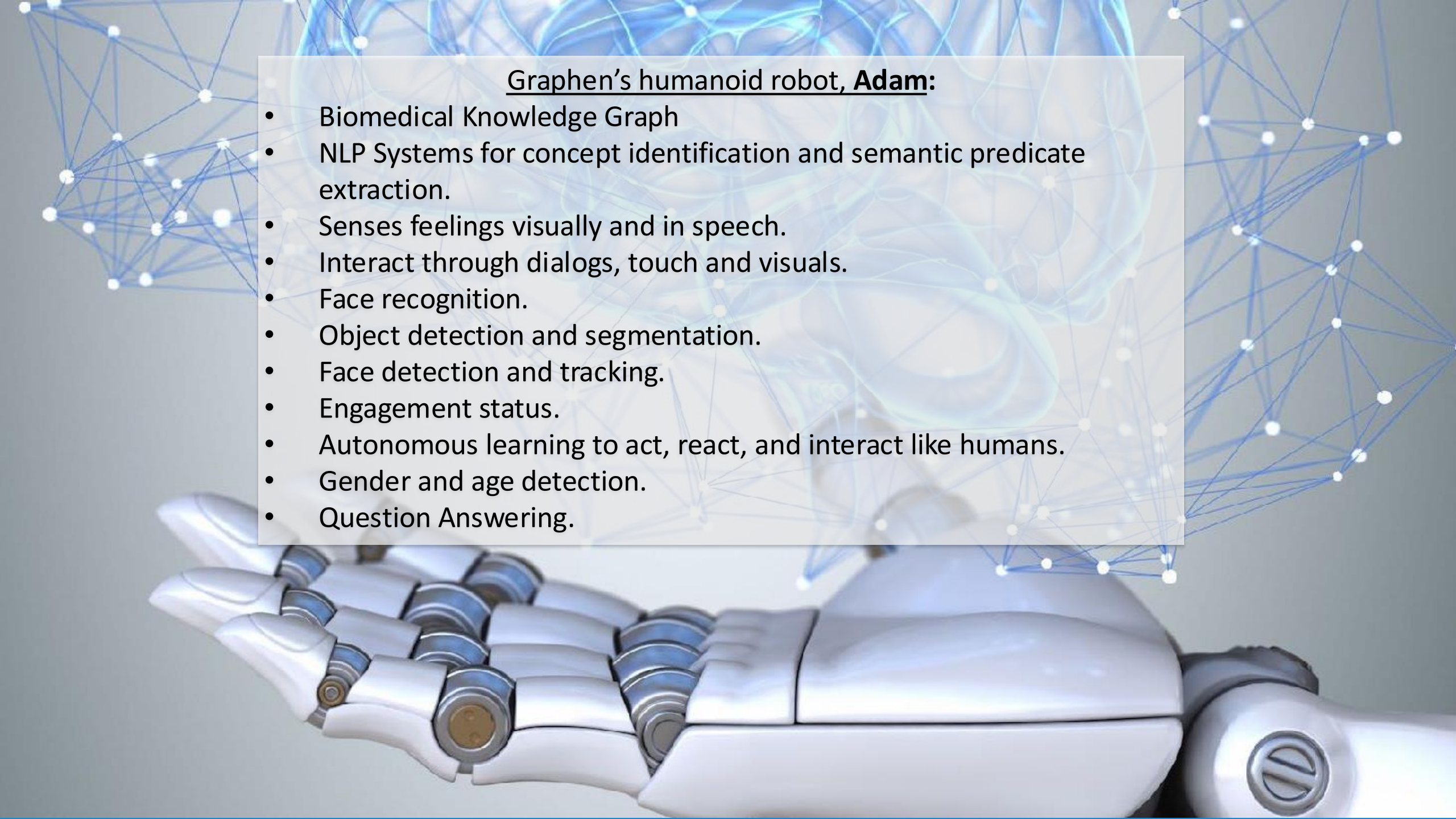


Graphen's Ardi Sense achieves Deep Video Understanding in the ACM Multimedia Grand Challenge (2nd place in 2020, 1st place in 2021 and 2022):

- Visual Recognition
- Speech Recognition
- Knowledge Graph
- Face Recognition
- Emotion Recognition
- Speaker Identification
- Relationship Inference
- Event and Action Understanding

Ardi Sense's Natural Language Understanding:

- Reading Medical Articles -> summarization & Q&A
- Reading Financial Information and Market Info

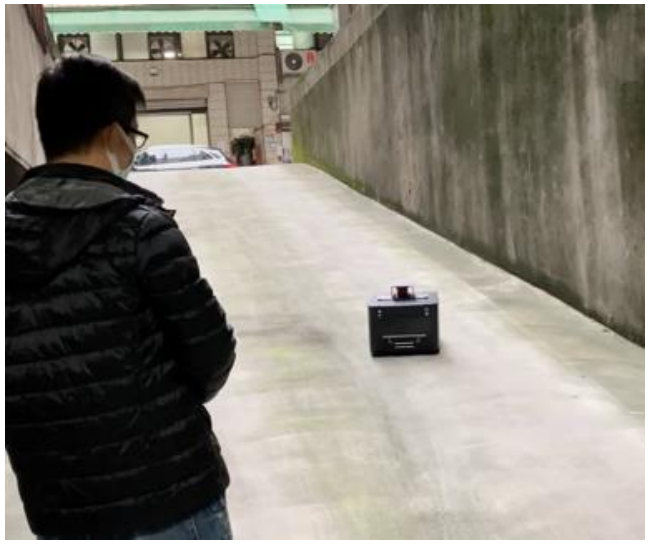


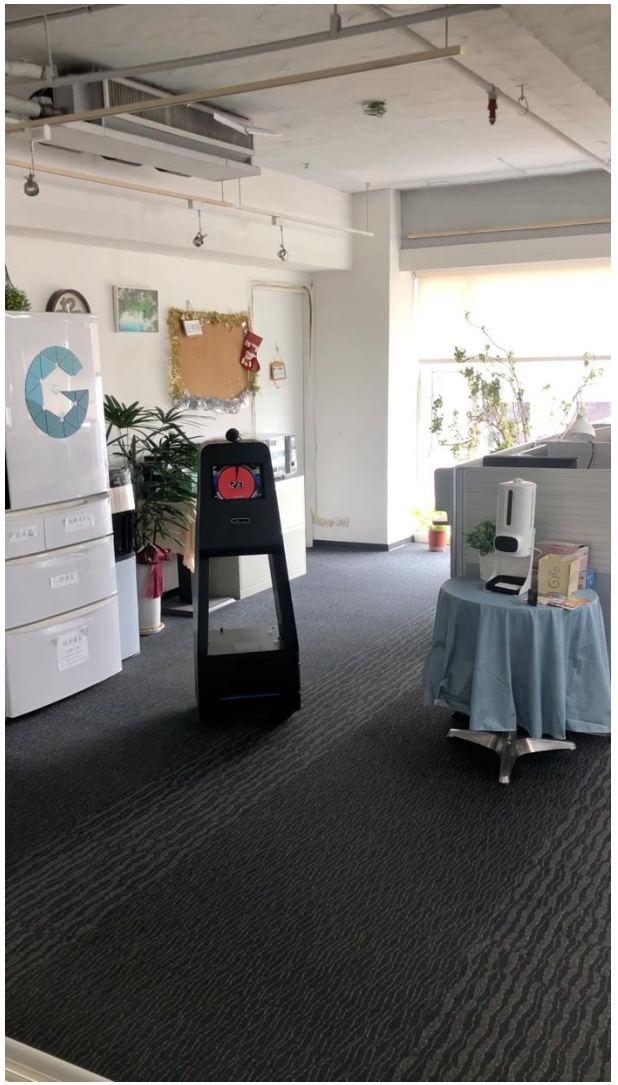
Graphen's humanoid robot, **Adam:**

- Biomedical Knowledge Graph
- NLP Systems for concept identification and semantic predicate extraction.
- Senses feelings visually and in speech.
- Interact through dialogs, touch and visuals.
- Face recognition.
- Object detection and segmentation.
- Face detection and tracking.
- Engagement status.
- Autonomous learning to act, react, and interact like humans.
- Gender and age detection.
- Question Answering.

Graphen Adam II

- Graphen created an Indoor/Outdoor Dual-Service Robot product.
- **Developed AI Computer Vision to replace LiDAR – big price and performance differentiator:**
  - **Computer Vision-based Cruising, Navigation, Map Scanning, etc.**
- **Developed AI Speech Interactions**
- **Can be applied to various industrial scenarios by replacing upper components**





New Graphen Robotics Hardware

Aiia Robot

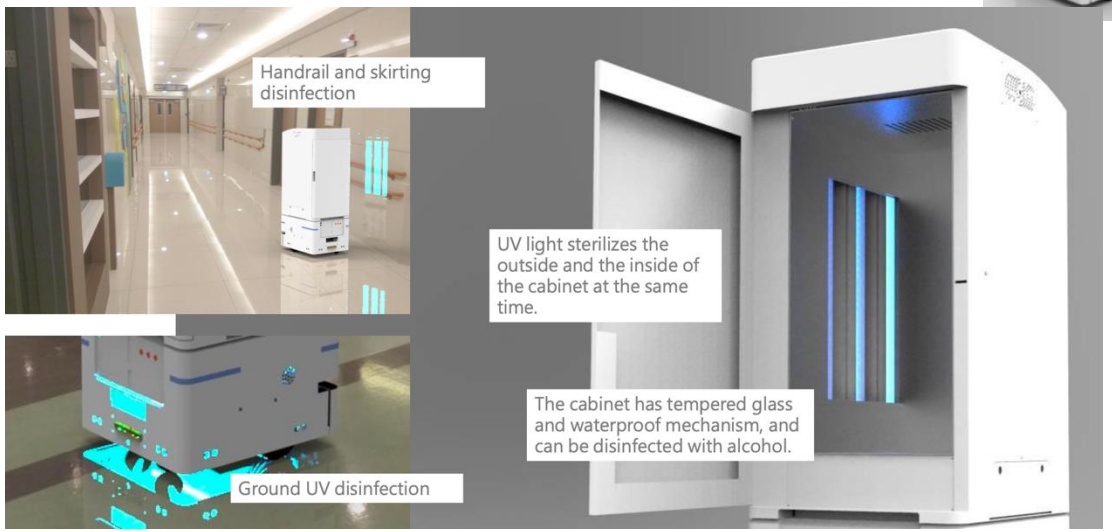
Aiia Robot-II

Robotic Arms

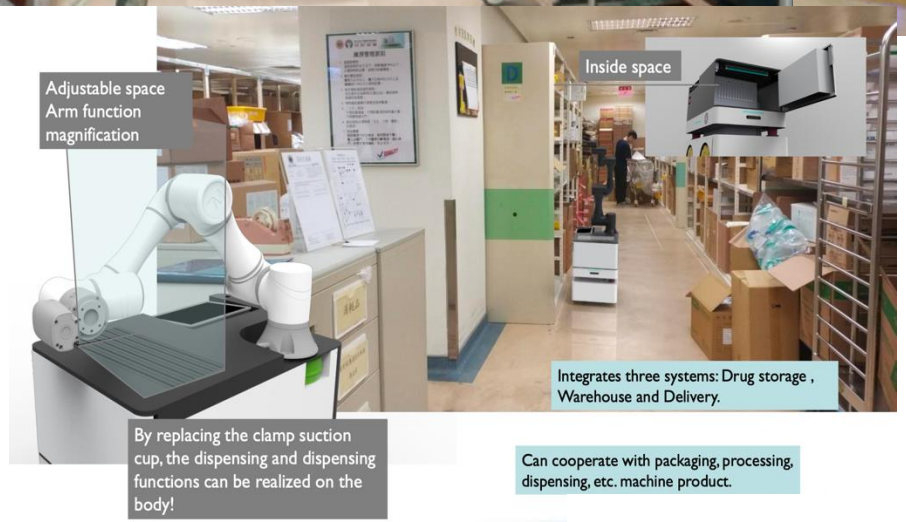
https://www.youtube.com/watch?v=MDDdPlWyiDw

Service Robots for Hospitals

Carrier Robot in Hospitals



Disinfection Robot



Autonomous Mobile Robot