BRECCIA: A Multi-Agent Data Fusion and Decision Support Framework for Dynamic Mission Planning

D. Sacharny, T.C. Henderson, A. Mitiche, R. Simmons, and T. Welker
School of Computing
University of Utah
Salt Lake City, UT, USA
Email: tch@cs.utah.edu

X. Fan
Nanyang Technological University
Singapore
Email: xyfan@ntu.edu.sg

***Abstract*— Geospatial Intelligence analysis involves the combination of multi-source information expressed in logical form (as sentences or statements), computational form (as numerical models of physics or other processes), and sensor data (as measurements from transducers). As systems become more complex, the uncertainty with any bit of entailed knowledge becomes more ambiguous and difficult to rationalize. This fact suggests that the core purpose for data-driven dynamic application systems is to support the goals of human agents, which ultimately includes making rational decisions, and therefore must integrate neatly within the human decision-making framework. Despite the fact that continuous models are the most accurate model of the environment, decision making is a discrete process that requires rational logic and a well-defined language to support communication between agents. The process of fusion of information from continuous and discrete models and applying decision-making is analogous to the cognitive process of humans, which suggests a cognitive framework for computation is warranted. We propose *BRECCIA*, a Geospatial Intelligence analysis and decision-support system, designed to support rational decision-making in continuous environments with the following characteristics: the fusion of discrete logical processes and continuous models, the ability to simulate courses of action that take into account real-time information, and the ability to automatically and continuously plan based on real-time information. In addition to the theoretical foundations underlying this system, we apply the framework to a particular scenario that includes planning and executing a simulated multi-agent UAV mission in an urban environment.**

I. CONTRIBUTIONS

The *BRECCIA* system is composed of multiple BDI agents (called PL-agents, for Probabilistic Logic), each of which is composed of a probabilistic inference module, detailed below. Agents are designed using an extension of the language AgentSpeak, called Jason [1]. Specializations of the PL-agent in the context of mission-planning may include:

- Mission-planner: responsible for constructing high-level plans such as reconnaissance missions.
- UAV-manager: responsible for managing the state and schedule of multiple unmanned air-vehicles (UAV).
- UAV: responsible for managing the state of a particular UAV.
- Weather-monitor: responsible for maintaining assumptions about the weather in named-areas-of-interest (NAOI) or along particular paths.
- User-manager: responsible for maintaining assumptions about particular users of the system. For example, whether a prior simulation's assumptions may have changed due to current information.

BRECCIA is designed to integrate with human agents by mirroring a simplified cognitive model, called the belief-desire-intention (BDI) model, and incorporating fast probabilistic logic to provide simple justifications for planned actions. We assert that probabilistic logic provides the framework for fusing uncertainty from continuous processes with uncertainty from discrete logical models. In addition to the fusion framework, a core requirement for adequate decision-support systems is the ability to simulate courses of action while integrating uncertainty about the justifications for such actions. The central contribution of this paper is the design of a system that supports such human decision-making.

The probabilistic logic module is a major contribution to this multi-agent design (for a formal reasoning behind using modules in BDI architectures, see [4]). Within the module, justifications and probabilities are assigned to PL-agent beliefs that are entailed from the knowledge base. Probabilities and justifications are attached to beliefs via annotations. For example, a belief that follows from beliefs about the weather may be represented as:

$$weather\_ok[p(0.86),\\ justification(mat\_imp(weather\_ok))]$$

This belief states that the weather is "OK" with probability 0.86 and with a justification that traces back to a material implication rule in the knowledge base. Custom rules in the knowledge base, such as the material implication referenced in the above justification, enable a special backward chaining in the probabilistic logic module. Probabilistic material implication rules are defined as in the following example:

$$mat\_imp(weather\_ok,\\ [visibility\_ok, temperature\_ok])[p(0.95)]$$

This rule states the following, in traditional logic: $(visibility\_ok \wedge temperature\_ok) \implies weather\_ok$, and that this rule holds with 0.95 probability. The backward chaining algorithm is shown in Algorithm 1. Critical to the implementation of Algorithm 1 is the use of the non-linear probabilistic logic (NLPL) function. This function represents

Data: $L \leftarrow$ List of antecedents
Result: $p(C)$ - The probability of the consequent
$matched\_beliefs = [empty\_list]$;
for *each A in L* **do**
  $belief \leftarrow match\_in\_kb(A)$;
  $p(belief) \leftarrow extract\_probability(belief)$;
  $append(matched\_beliefs, belief[p(belief)])$;
end
$p(C) = run\_nlpl(matched\_beliefs)$;
add $C[p(C), justification(mat\_imp(C)]$ to KB;

Algorithm 1: Probabilistic Backward Chaining

Data: $L \leftarrow$ List of justifications
Result: Reduced Uncertainty
$matched\_beliefs = []$;
for *each A in L* **do**
  $belief \leftarrow match\_in\_kb(A)$;
  $p(belief) \leftarrow extract\_probability(belief)$;
  $append(matched\_beliefs, belief[p(belief)])$;
end
$!reduce\_uncertainty(min(p(matched\_beliefs)))$
$p(C) = run\_nlpl(matched\_beliefs)$;
revise $C[p(C), justification(mat\_imp(C)]$ in KB;

Algorithm 2: Uncertainty Reduction

a novel approach to fusing probabilities of propositions, and is detailed in [2].

At the heart of probabilistic logic is the notion of worlds [3], the concept of which has been explored as a unifying element of knowledge representation in [5]. *BRECCIA* utilizes the notion of worlds in two aspects, first in the context of probabilistic logic, and second in the context of simulations.

BRECCIA agents are capable of reasoning about beliefs in multiple worlds, or more concretely, namespaces [4], to facilitate simulations by users and eventually autonomous agents. The experimental simulation, detailed in the results section, is executed by a user that endows the *BRECCIA* agents with particular beliefs about a particular world.

Probabilistic logic represents a high-level form of data fusion because uncertainty associated with low-level functions are translated into the discrete domain of propositional logic. For example, mission success may be defined by a multitude of factors and one of which may include whether an unmanned air-vehicle is able to reach a destination by a specific time. That determination may depend on the weather, which entails a certain probability. *BRECCIA* agents that specialize in the domain of weather, for instance, execute plans to gather data and translate it into logic statements with probability. Finally a human agent may query the system for mission success to obtain a high-level interpretation of all the mission's parameters.

BRECCIA embraces a data-driven approach by connecting belief-revision to plans and actions. As a concrete example, consider a user of *BRECCIA* that has completed a number of simulations under the assumption that visibility is good with probability 0.9. This fact is represented in the user agent as follows:

$$visibility(good)[p(0.9), source(weather\_monitor)]$$

An associated data-driven plan, executed when certainty has fallen below 0.6, to notify the user that their simulations may be invalidated due to changed assumptions, is represented as follows:

$$+visibility(good)[p(X)] : X < 0.6 \leftarrow !tell\_user$$

This concise representation of data-driven planning is the hallmark of reactive systems. In addition to the reactive nature of *BRECCIA*, the system also exhibits proactive behavior by embracing the goal-driven nature of the BDI architecture. Our main proactive component is our algorithm for uncertainty reduction. Agents in *BRECCIA* may be programmed to achieve this particular goal in the following syntax:

$$!reduce\_uncertainty(proposition)$$

Algorithm 2 shows how *BRECCIA* currently reduces uncertainty by recursively finding the minimum probability in a dependency tree and adding the goal to reduce uncertainty for that belief. Contingency plans may also be executed in the case that the minimum probability belongs to a base assumption that cannot change.

II. Results

A simulated mission scenario containing a number of base assumptions about the world is executed on the Jason framework. There are 23 sentences distributed across three agents (mission_planner, uav_manager, and weather_monitor). Material implication rules are each given a probability of 0.9, while ground atoms all have probabilities of 0.8. The material implication rules are as follows (the agent is shown as a predicate to the rule):

$$\begin{aligned}
&mission_planner(mat_imp(collection_done,\\
&\quad [raven_infra_red, target_loiter_ok])),\\
&mission_planner(mat_imp(path_ok,\\
&\quad [raven_power_ok, raven_battery_ok\\
&\quad speed_known, altitude_known,\\
&\quad loiter_time_known,\\
&\quad route_time_known, naoi_defined])),\\
&mission_planner(mat_imp(mission_ok,\\
&\quad [raven_platform_available,\\
&\quad raven_air_control_ok, weather_ok,\\
&\quad collection_done, path_ok,\\
&\quad crew_available, air_defense_known])),\\
&weather_monitor(mat_imp(weather_ok,\\
&\quad [wind_under_17, precip_low,\\
&\quad visibility_ok, temperature_ok]))
\end{aligned}$$

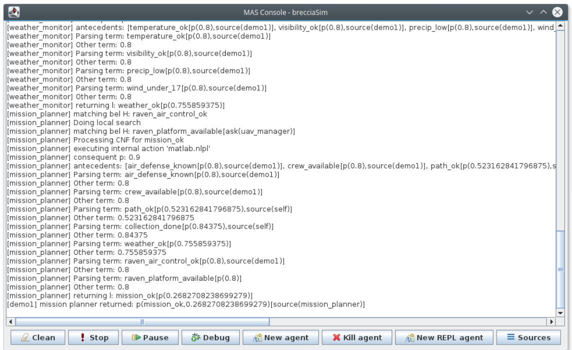

Fig. 1. *BRECCIA* Mission Simulation Output

The simulation is run by a user agent with the following plan:

$+!run\_sim <-$

$.print($"Running simulation"$);$

$.send(mission\_planner, askOne,$

$p(mission\_ok, X), A);$

$.print($"mission planner returned: ", A$);$

$.send(mission\_planner, achieve,$

$reduce\_uncertainty(mission_ok, Y), Z);$

$.print($"Uncertainty reduction produced result:", A$);$

BRECCIA finds the following values for the implicit atoms: $p(weather\_ok) = 0.7559$, $p(collection\_done) = 0.84375$, $p(path\_ok) = 0.5232$, and $p(mission\_ok) = 0.2683$. The uncertainty reduction locates the lowest probability, $path\_ok$ and executes a simulated plan to increase it. The simulated plan increases the probability of $path\_ok$ to 0.85 by increasing each of its dependent atom probabilities to 0.95. The final query probability is resolved to $p(mission\_ok) = 0.5532$. Figure 1 shows the console output from the simulation.

III. Acknowledgment

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-17-1-0077.

References

[1] R.H. Bordini, J.F. Huebner, and M. Wooldridge. *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley Series in Agent Technology. Wiley, Hoboken, NJ, 2007.

[2] T.C. Henderson, A. Mitiche, R. Simmons, and X. Fan. A Preliminary Study of Probabilistic Argumentation. Technical Report UUCS-17-001, University of Utah, February 2017.

[3] N. Nilsson. Probabilistic Logic. *Artificial Intelligence Journal*, 28:71–87, 1986.

[4] G. Ortiz-Hernández, J.F. Hübner, R.H. Bordini, A. Guerra-Hernández, G.J. Hoyos-Rivera, and N. Cruz-Ramírez. A Namespace Approach for Modularity in BDI Programming Languages. In M. Baldoni, J.P. Müller, I. Nunes, and R. Zalila-Wenkstern, editors, *Engineering Multi-Agent Systems: 4th International Workshop, EMAS 2016, Singapore, Singapore, May 9-10, 2016, Revised, Selected, and Invited Papers*, pages 117–135, Cham, 2016. Springer International Publishing.

[5] Jonathan R. Scally, Nicholas L. Cassimatis, and Hiroyuki Uchida. Worlds as a unifying element of knowledge representation. *Biologically Inspired Cognitive Architectures*, 1:14–22, July 2012.