Computational Models of Argument
C. Reed et al. (Eds.)
© 2024 The Authors.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA240344

ABA Argument Graphs with Constraints

Jeff THOMPSON[a,1] and Kristinn R. THÓRISSON[a, b]
[a]*Icelandic Institute for Intelligent Machines (IIIM), Reykjavík, Iceland*
[b]*Center for Analysis & Design of Intelligent Agents, Reykjavik University, Iceland*
ORCiD ID: Kristinn R. Thórisson https://orcid.org/0000-0003-3842-0564

1. Introduction

A central hypothesis in artificial general intelligence is that it requires flexible reasoning. Our research on the AGI-aspiring system AERA[2] (Autocatalytic Endogenous Reflective Architecture) [2,3] couples argumentation with transparent causal reasoning, allowing for autonomous self-explanation. Craven and Toni [1] describe ABA[3] argument graphs and a derivation procedure for computing them. They also provide the *abagraph* software which implements the derivation procedure. The proof procedure and abagraph software are only defined for ground terms. But, as described below, the domain-general learning in AERA require a more flexible solution. We extend the abagraph software with explicit variables and constraints on them.[4] Here we describe why we need variables with constraints and how argumentation is implemented and used by AERA at runtime.

2. Argumentation in AERA

Like most argumentation frameworks, ABA requires a defined logic language. AERA is a real-time architecture for learning, knowledge representation and reasoning in a cognitive agent (e.g. a robot), where argumentation is used for self-explanation [4]. Because it is for real-time control, time is explicitly represented in AERA. Its logic language specifies that a fact holds in a time interval, for example *fact(position(h, 25), 400, 500)* means that a robot hand *h* had position 25 within the time interval 400 to 500 milliseconds. It also represents events, for example, *fact(move(h, 10), 400, 500)* means that the command was issued to move *h* by a distance of 10 in this interval.

AERA uses rules to describe how part of the system changes from one state to another as a causal result of an action. While ABA allows rules of any form (as long as the conclusion is not an assumption), AERA uses more specific rules, for example *fact(position(h, 35), 500, 600)* $\leftarrow$ *fact(position(h, 25), 400, 500), fact(move(h, 10), 400, 500)* means that if *h* has position 25 and receives the command to move by a distance of 10 (in the time interval 300 to 400) then this causes *h* to have position 35 in the later time

[1] Corresponding Author: Jeff Thompson, jeff@iiim.is .

[2] See http://OpenAERA.org and http://github.com/IIIM-IS/AERA .

[3] *Assumption-based argumentation*, implemented at http://robertcraven.org/proarg/abagraph.html .

[4] While abagraph is implemented in Prolog, and in some cases it's possible to use Prolog variables with late binding, the ABA graph proof procedure was not designed to be used with Prolog variables and they do not directly support constraints, which are needed for AERA reasoning.

interval 400 to 500. We view the conclusion of the rule as a goal state and reasoning proceeds through backward chaining to find a solution.

3. Implementing Constraints

Using rules with ground values for state variables (like position) and specific object identifiers (like *h*) is not sufficiently general. The current work extends ABA to support variables which have the form *var(n)* where *n* is an integer ID, and adds a constraint store to the derivation tuple. We can now state *fact(position(var(1), 35), 500, 600)* which is the goal to have some hand *var(1)*[5] (depending on availability in the environment) at position 35. We also use variables for the position and add a constraint: *fact(position(var(1), var(2)), 500, 600) ← fact(position(var(1), var(5)), 400, 500), fact(move(var(1), var(6)), 400, 500), var(2) = var(5) + var(6)* means that if the hand is at initial position *var(5)* and moves by a distance *var(6)* then it will have final position *var(2)* where the final position is the initial position plus distance moved.[6]

In **Figure 1**, the AERA Visualizer[7] shows a "freeze frame" in the derivation. When a variable is assigned a ground value in the constraint store, the related variables in constraints can be evaluated. In this case, the distance moved *var(6)* should be 10.0.

Boolean constraints are also supported, for example *var(6) < 50* means the distance moved must be less than 50. If this is violated when the constraint store evaluates *var(6)* then (this branch of) the derivation fails.

With variables and constraints, ABA is more expressive and can support a real-time system like AERA.

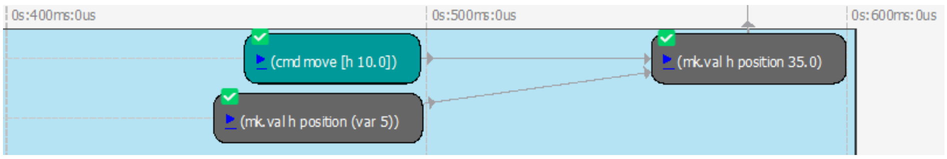

Figure 1. AERA Visualizer with constraint variables partially resolved.

References

[1] Craven R, Toni F. (2016). Argument graphs and assumption-based argumentation. Artificial Intelligence, **233**:1-59.
[2] Thórisson, K.R.(2020). Seed-Programmed Autonomous General Learning. In: Proceedings of Machine Learning Research, **131**:32–70.
[3] Thórisson, K.R., Rörbeck, H., Thompson, J. & Latapie, H. (2023). Explicit Goal-Driven Autonomous Self-Explanation Generation. In: P. Hammer, M. Alirezaie & C. Strannegard (eds.), Proc. Artif. General Intell., 826-295.
[4] Thórisson K.R. (2021). The 'Explanation Hypothesis' in General Self-Supervised Learning. Proc. Machine Learning Research, **159**:5-27.

[5] From the point of view of the ABA derivation procedure, *var(1)* is a ground term.
[6] The demo also shows variables for the time interval like 400 to 500 (omitted here for brevity).
[7] See https://github.com/IIIM-IS/AERA_Visualizer .