Behavior Trees with Dataflow: Coordinating Reactive Tasks in Lingua Franca

Alexander Schulz-Rosengarten
Akash Ahmad
Malte Clement
Reinhard von Hanxleden
{als,stu222517,mac,rvh}
@informatik.uni-kiel.de
Kiel University
Kiel, Germany

Benjamin Asch
Marten Lohstroh
Edward A. Lee
{benjamintasch,marten,eal}
@berkeley.edu
UC Berkeley
Berkeley, USA

Gustavo Quiros
Ankit Shukla
{gustavo.quiros,ankit.shukla}
@siemens.com
Siemens Technology
USA

ABSTRACT

Behavior Trees (BTs) provide a lean set of control flow elements that are easily composable in a modular tree structure. They are well established for modeling the high-level behavior of non-player characters in computer games and recently gained popularity in other areas such as industrial automation.

While BTs nicely express control, data handling aspects so far must be provided separately, e. g. in the form of blackboards. This may hamper reusability and can be a source of nondeterminism.

We here propose a dataflow extension to BTs that explicitly models data relations and communication. We realize and validate that approach in the recently introduced polyglot coordination language Lingua Franca (LF).

CCS CONCEPTS

• **Software and its engineering** → **Model-driven software engineering**; *Abstraction, modeling and modularity*; *Visual languages*; **Orchestration languages**.

KEYWORDS

Behavior trees, reactive systems, coordination languages

ACM Reference Format:
Alexander Schulz-Rosengarten, Akash Ahmad, Malte Clement, Reinhard von Hanxleden, Benjamin Asch, Marten Lohstroh, Edward A. Lee, Gustavo Quiros, and Ankit Shukla. 2024. Behavior Trees with Dataflow: Coordinating Reactive Tasks in Lingua Franca. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3639478.3643093

1 INTRODUCTION

BTs originated in the gaming industry, where they are used to program non-player characters [1]. They express complex behavior with highly reactive and modular software components coordinating agents in groups. Their simplicity and modularity has made them increasingly popular in real-world applications as well, such as industrial automation, where BTs control machines and robots in automated factories. BTs use a model-based approach with a simple and intuitive tree structure and a lean set of control flow elements.

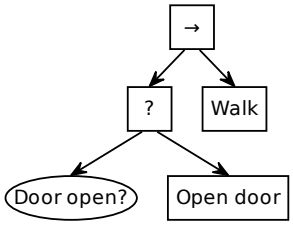

Figure 1: BT illustrating sequence, fallback, conditional, and task nodes.

Fig. 1 illustrates a simple example behavior modeled as an BT. If the door is closed, it should be opened before walking through. The root node is a *sequence* node (→) that executes its children sequentially as long as they return SUCCESS. If there is no further child to execute, the sequence node returns SUCCESS. If a child returns RUNNING or FAILURE, the sequence stops there and immediately returns RUNNING or FAILURE. The first child of the sequence is a *fallback* node (?), which is symmetric by sequentially executing child nodes iff the previous node returns FAILURE. Its first child is a *condition* node (ellipsis) that checks if the door is already open. If closed, it will report a FAILURE, which issues the execution of the *task* node Open door. After the fallback reports SUCCESS, either due to the condition or the task succeeding, the sequence will start the Walk task. There is also a *parallel* node in BTs that executes children concurrently, which is not illustrated in this example.

1.1 Motivation

While the simplicity of BTs is attractive, its minimalist notation leaves the aspect of *handling data* unaddressed. This aspect, however, is crucial when using BTs for implementing behaviors that need to adapt based on data. In cyber-physical systems, such as robots or of vehicles, data may originate from sensors that inform the software about the state of its environment. A common solution is to use a *blackboard* [3] that introduces a global set of variables to a BT. However, in combination with a parallel composition in a BT, unconstrained access to shared variables can easily lead to race conditions and non-deterministic behavior. This hampers reproducibility, robustness, and debugging, and may be fatal when designing safety-critical software.

ICSE-Companion '24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0502-1/24/04
https://doi.org/10.1145/3639478.3643093

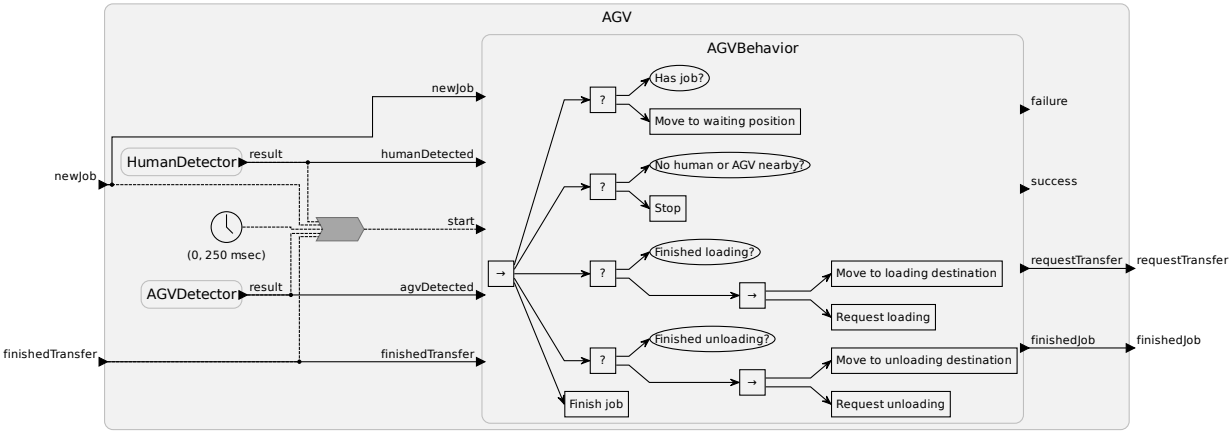

Figure 2: A reactor for controlling an Automated Guided Vehicle (AGV) including an BT for specifying its behavior.

2 APPROACH

To address the handling of data, we propose to adopt a dataflow notation. This notation represents a system as blocks with streams of data flowing between them, as illustrated by actor models or the SCADE tool. Dataflow, unlike blackboards, explicitly models data interfaces and communication, thus supporting modularity. Dataflow nodes, or *actors*, have explicit inputs and outputs, which define the way instances of these nodes need to be interconnected with their surrounding context. BTs provide a modular design, and instantiating a node or entire BT as a child seems rather straightforward, which facilitates reusability. However, if such a node or BT relies on access to a blackboard and shares data with other nodes, this constitutes a rather brittle interface. There is typically no indication which variables are considered inputs, outputs, or only local, and how to address multiple instantiations and their memory. We propose to treat communication as a first-class citizen, as modeling it explicitly facilitates formal analyses and program comprehension.

We use *Lingua Franca (LF)* [2] as general setting, since we consider LF and its open-source infrastructure a good match to address the aforementioned problems. LF is rooted in *reactors*, a reactive, event-based, timed-sensitive, and concurrent model of computation with deterministic semantics. LF is a polyglot coordination language in which reactors encapsulate reactive tasks specified in verbatim code. Reactors provide a high-level *coordination layer* to orchestrate the execution of complex software systems, similar to BTs coordinating the execution of nodes. This coordination layer follows a dataflow notation, interconnecting reactor instances. The LF framework also provides advanced modeling capabilities and tooling for automatically generating customized graphics from a textual file, e. g., Fig. 2 has been synthesized this way.

3 BEHAVIOR TREES IN LINGUA FRANCA

The basic idea of BTs in LF is that a BT should be a new kind of reactor, whose inner behavior is coordinated by a BT structure instead of a classical reactor composition. A BT reactor should be admissible wherever a normal reactor can be used in an LF program.

This integration is illustrated by the diagram in Fig. 2. It shows a reactor that controls an Automated Guided Vehicle (AGV), which we present in full detail in [4]. The reactor composes a timing element (the clock node), a reaction (the gray wedge) that processes events, and other reactors, such as HumanDetector that handles sensor data processing. At the same time, our concept allows reactors specified using the BT notation, as in AGVBehavior. Our approach also includes a visualization and transformation that maps the BT structure directly into LF and thus introduces the well-formed and data-oriented LF semantics into the BT notation [4].

4 CONCLUSION

Our proposal on augmenting BTs with dataflow is, to our knowledge, the first attempt to do so systematically at the level of a coordination language. The aim is to combine the best of two worlds that, so far, have seen little interaction through the involved research communities or in actual practice. We argue that these concepts can be of mutual benefit. Compared to ordinary BTs, our approach improves modularity and ensures determinism by replacing rather unstructured blackboards with a clean dataflow notation. Conversely, dataflow formalisms can harness the intuitive, compact BT machinery that by now is proven in practice in a large and still growing community of users in game development, robotics control, industrial automation, etc.

With LF as the basis for a concrete realization of our proposal, we leverage its deterministic semantics for concurrent, distributed real-time systems. Moreover, LF's polyglot nature makes our proposal compatible with a wide range of target languages.

REFERENCES

[1] Michele Colledanchise and Petter Ögren. 2018. *Behavior Trees in Robotics and AI: An Introduction.* CRC Press. https://doi.org/10.1201/9780429489105
[2] Marten Lohstroh, Christian Menard, Soroush Bateni, and Edward A. Lee. 2021. Toward a Lingua Franca for Deterministic Concurrent Systems. *ACM Transactions on Embedded Computing Systems (TECS)* 20, 4 (May 2021), Article 36. https://doi.org/10.1145/3448128
[3] Alejandro Marzinotto, Michele Colledanchise, Christian Smith, and Peter Ögren. 2014. Towards a unified framework for robot control. In *2014 IEEE International Conference on Robotics and Automation (ICRA).* 5420–5427. https://doi.org/10.1109/ICRA.2014.6907656
[4] Alexander Schulz-Rosengarten, Akash Ahmad, Malte Clement, Reinhard von Hanxleden, Benjamin Asch, Marten Lohstroh, Edward A. Lee, Gustavo Quiros Araya, and Ankit Shukla. 2024. Behavior Trees with Dataflow: Coordinating Reactive Tasks in Lingua Franca. https://doi.org/10.48550/ARXIV.2401.09185