An Empirical Study of Agent Programs
A Dynamic Blocks World Case Study in GOAL

M. Birna van Riemsdijk and Koen V. Hindriks
EEMCS, Delft University of Technology, Delft, The Netherlands
{m.b.vanriemsdijk,k.v.hindriks}@tudelft.nl

Abstract. Agent-oriented programming has been motivated in part by
the conception that high-level programming constructs based on common
sense notions such as beliefs and goals provide appropriate abstraction
tools to develop autonomous software. Various agent programming lan-
guages and frameworks have been developed by now, but no systematic
study has been done as to how the language constructs in these lan-
guages may and are in fact used in practice. Performing a study of these
aspects may contribute to the design of best practices or programming
guidelines for agent programming, and clarify the use of common sense
notions in agent programs. In this paper, we analyze various agent pro-
grams for dynamic blocks world, written in the GOAL agent programming
language. We present several observations based on a quantitative and
qualitative analysis that provide insight into more practical aspects of
the development of agent programs. Finally, we identify important is-
issues in three key areas related to agent-oriented programming that need
further investigation.

1 Introduction

The concept of a goal lies at the basis of our understanding of why we perform
actions. It is common sense to explain the things we do in terms of beliefs and
goals. The reasons for performing actions are derived from our motivations and
the notion of rational behavior is typically explained in terms of actions that are
produced in order to further our goals. For example, a researcher who has a goal
to have finished a paper but is going on a holiday instead is not considered to
behave rationally because holidays do not further the goal of writing a paper.

Shoham was one of the first who proposed to use such common sense notions to
build programs [14], coining a new programming paradigm called agent-oriented
programming. Inspired by Shoham, a variety of agent-oriented programming lan-
guages and frameworks have been proposed since then [314]. For several of them,
interpreters and Integrated Development Environments (IDEs) are being devel-
oped. Some of them have been designed mainly with a focus on building prac-
tical applications (e.g., JACK [17] and Jadex [13]), while for others the focus
has been also or mainly on the languages’ theoretical underpinnings (e.g., GOAL
[10], 2APL [6], and Jason [5]).
In this paper, we take the language GOAL as our object of study. GOAL is a high-level programming language to program rational agents that derive their choice of action from their beliefs and goals. Although the language’s theoretical basis is important, it is designed by taking a definite engineering stance and aims at providing useful programming constructs to develop agent programs.

However, although it has been used for developing several small-size applications, no systematic study has been done as to how the language constructs are used in practice to program agents, and how easy it is to read the resulting programs. Also, if the language is going to be used for building real-world applications, efficiency and a programmer’s knowledge of this becomes an important issue. We believe it is important to get a better understanding of these issues in order to identify which aspects need to be addressed with respect to them, and to be able to design a set of best practices or programming guidelines that support GOAL programmers. It is the purpose of this paper to contribute to this aim. We do this by analyzing three GOAL programs for the dynamic blocks world domain. To the best of our knowledge, this is the first time such comparative analysis of agent programs programmed in a dedicated agent programming language has been done. This kind of empirical software engineering is expected to form over time a body of knowledge leading to widely accepted and well-formed theories.

The dynamic blocks world domain is explained in Section 2 and in Section 3 we explain the GOAL language. We outline our approach in Section 4. In Section 5, we analyze the programs using various numeric measures on their code and execution. In Section 6, we look in more detail at how the programs were written and how the language constructs of GOAL were used. We conclude in Section 7.

2 The Dynamic Blocks World

The blocks world is a simple environment that consists of a finite number of blocks that are stacked into towers on a table of unlimited size. It is assumed that each block has a unique label or name a, b, c, .... Blocks need to obey the following “laws” of the blocks world: (i) a block is either on top of another block or it is located somewhere on the table; (ii) a block can be directly on top of at most one other block; and, (iii) there is at most one block directly on top of any other block.

A blocks world problem is the problem of which actions to perform to transform an initial state or configuration of towers into a goal configuration, where the exact positioning of towers on the table is irrelevant. A blocks world problem thus defines an action selection problem. The action of moving a block is called constructive (see, e.g., [15]) if in the resulting state that block is in position, meaning that the block is on top of a block or on the table and this corresponds with the goal state, and all blocks (if any) below it are also in position. Observe that a constructive move always increases the number of blocks that are in position.

We have used a specific variant of the blocks world, which we call the dynamic blocks world. In the dynamic blocks world, a user can move blocks around while