A Software Package (*in progress*) that implements the Hammock-EFL Methodology

Moshe Goldstein[1][0000-0002-0429-4873] and Oren Eliezer[1][0000-0001-9699-8247]

[1] Jerusalem College of Technology – Lev Academic Center, Jerusalem 9372115, Israel
goldmosh@g.jct.ac.il orene10@gmail.com

Abstract. This poster paper presents a software package (*in progress*) that implements the Hammock-EFL approach for Project Management and Parallel Programming, written in Python.

Keywords: Hammock-EFL, Project Management, Parallel Programming, Python Programming

1 Motivation

In sequential programming a problem is solved by decomposing it into sub-problems and by identifying the structural dependencies among them. In project design and management, the designer needs to identify not only all the activities (or sub-problems) that compose the whole project and their structural dependencies, but also their *temporal* dependencies. The identification of those temporal dependencies implicitly requires the application of parallel thinking. The same is required from a programmer when he tries to solve a problem by Parallel Programming. All those problem-solving-related observations induced us to realize that a project management methodology, like the Hammock Cost Techniques for Project Management [1], will contribute to better problem solving thinking in Parallel Programming.

EFL (Embedded Flexible Language) [2,3] is a mini embedded language whose semantics are those of the Flexible Algorithms (FA) [4] approach to computation, which is applicable for parallel programming (as well as sequential programming) and ensures deterministic results without the need of locking. EFL was designed to make parallel programming independent of any specific parallel programming platform, making the programmer's task easier. To allow that independence, two EFL pre-compilers were implemented for the Python programming language as the host language.

Based on all the above, the Hammock-EFL methodology [5-7] was proposed. It combines the Hammock methodology and the FA approach to Computation. This combination allows developers to treat programs and project schedules as conceptually the same, at a higher level of abstraction, enabling them to deal with the complexity of computing systems engineering in a more reliable and easier way. This is *the* novelty of the Hammock-EFL methodology. We argue that this is the first research which makes a symbiotic combination of methodologies taken from two different disciplines - (Parallel) Computing and Programming, and Project Management.

2 Proof of Concept

The Project Schedule diagram depicted in Fig. 1 was used as the Proof of Concept of the combined methodology.

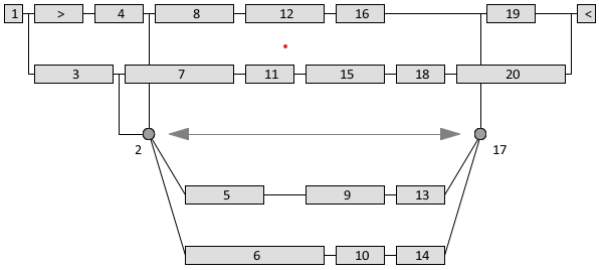

Fig. 1. The Project Schedule diagram (taken from [1]) used to try the software.

That schedule is composed by an appropriate combination of Regular, Compound and Hammock activities: two Compound activities are activated in parallel, the Regular activities that compose each of them, the activities no. 4, 8, 12, 16, and 19, and no. 3, 7, 11, 15, 18, and 20, are activated serially, and a Hammock activity H_1 whose Hammock members are organized in two groups, Regular activities no. 5, 9, and 13, and no. 6, 10, and 14. Each activity may include properties such as duration (D_i), earliest beginning (ES_i), latest beginning (LS_i), amount of resources (R_i), etc.

If the same diagram is intended to describe a program, a Regular activity represents a function with a relatively simple behavior, a Compound activity represents a function whose behavior is expressed by an appropriate combination of the behaviors of its sub-activities, a Hammock activity represents a function whose behavior is expressed by a randomly scheduled ensemble of sub-activities, executed in sequence. Based on the structural and temporal dependencies of the sub-activities of a Compound activity, the functions represented by them may be executed in parallel or in sequence.

The software presented here uses a tree data structure to represent project schedules like that in Fig. 1. Fig. 2 shows the rendering of the tree representation of that schedule.

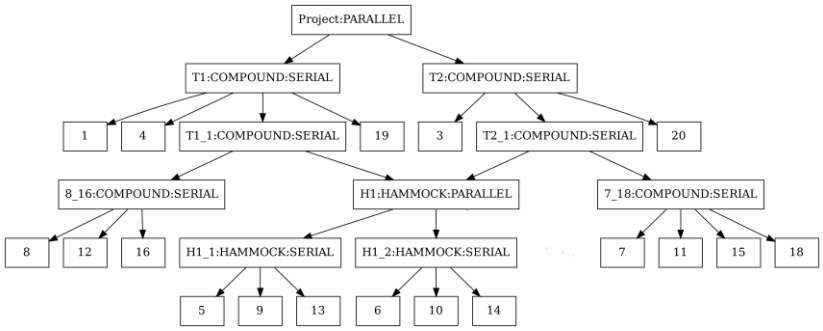

Fig. 2. The tree representation of the above project schedule

3 The Software

Fig. 3 shows a graph of the modules that compose the software presented here, and their dependencies. Table 1 gives a brief description of each one of those modules.

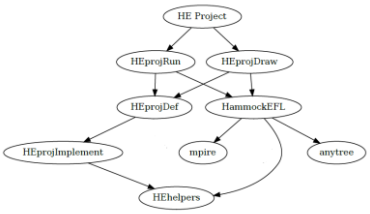

Fig. 3. The Dependency Graph of the Software.

Table 1. Description of modules that compose the software.

Program/Module	Description
anytree	A Python module implementing tree data structures.
HammockEFL.py	*Schedule* and *Activity* are classes defined in this module. See [7].
HEhelpers.py	Helper functions are defined for general use, as needed.
HEprojRun.py	Module used to actually run or simulate the schedule.
HEprojDef.py	A tree data structure, representing a schedule (or program), will be defined by the designer (or programmer), using methods from the *Schedule* and *Activity* classes.
HEprojDraw.py	A schedule's diagram is rendered using this module (see Fig. 2).
HEprojImplement.py	The behavior of each activity is expressed by a three-step (pre-processing → processing → post-processing) pipeline-like procedure. This module is intended to include their implementations.
mpire	A module that substitutes Python's Multiprocessing module.

4 Experiment

The software was successfully tried to run and simulate the above Proof of Concept schedule, restricted to the case that all the activities are defined to be executed in sequence. At the current moment, we are debugging the capability of running and simulating schedules that combine activities defined to be executed in sequence, with activities defined to be executed in parallel.

5 Conclusion and further work

The current state of a software package has been presented, which intends to be both, a framework for project management based on the Hammock Cost Techniques, and for

developing parallel software based on the FA approach to Computation. Following [1], the impact of the presented tool should be expressed as a trade-off between project's makespan and total revenue. To check this, a comparative and experimental research must be done when D_i, ES_i, LS_i, R_i and cost, for each activity i, will be taken into account in the case of a planned project relative to an already executed project. A new version of the software should include (a) a shared-memory mechanism like PSTM [8] which will allow the implementation of shared-memory-based parallel programs, and (b) support of JSON files (or YAML files) which will allow a more readable and writable definition of a schedule (or program). Additionally, experimental analysis of Time and Space Complexity of a designed computing system will be possible by calculating actual run time and storage of programs described by graphs of the kind discussed above.

References

1. G. Csébfalvi, A. Csébfalvi.: Hammock activities in Project scheduling. In Proceedings of the Sixteenth Annual Conference of POMS, POMS, Chicago, IL, USA (2005).
2. D. Dayan, M. Goldstein, M. Popovic, Sh. Mizrahi, M. Rabin, D. Berlovitz, O. Berlovitz, E. Bussani Levy, M. Naaman, M. Nagar, D. Soudry, R. B. Yehezkael.: EFL: Implementing and Testing an Embedded Language Which Provides Safe and Efficient Parallel Execution. In: Proceedings of ECBS-EERC 2015, pp. 83-90. IEEE Press, Brno, Czech Republic (2015). DOI: 10.1109/ECBS-EERC.2015.21.
3. M. Goldstein, D. Dayan, M. Rabin, D. Berlovitz, O. Berlovitz, R. B. Yehezkael.: Design principles of an embedded language (EFL) enabling well defined order-independent execution. In: Proceedings of ECBS 2017, pp. 1-8. ACM, Larnaca, Cyprus (2017). DOI: 10.1145/3123779.3123789.
4. R. B. Yehezkael, M. Goldstein, D. Dayan, Sh. Mizrahi.: Flexible Algorithms: Enabling Well-defined Order-Independent Execution with an Imperative Programming Style. In: Proceedings of ECBS-EERC 2015, pp. 75–82. IEEE Press, Brno, Czech Republic (2015). DOI: 10.1109/ECBS-EERC.2015.20.
5. O. Eliezer, M. Goldstein.: Implementing Hammock Cost Techniques using the parallel programming paradigm of EFL (Embedded Flexible Language). In: Procedings of ZINC 2018, pp 132-134, IEEE Press, Novi-Sad, Serbia (2018). DOI: 10.1109/ZINC.2018.8448763.
6. O. Eliezer, M. Goldstein, D. Dayan.: About the trade-off between time and space consumption when combining the Hammock-Cost model with the EFL (Embedded Flexible Language) parallel programming paradigm. In: Proceedings of the 16th International Conference on Civil, Structural and Environmental Engineering Computing, Riva del Garda, Italy (2019).
7. M. Goldstein, O. Eliezer, D. Dayan.: Implementing the Hammock-EFL Methodology for Project Management and Parallel Programming. In: Proceedings of ECBS 2021, pp 1-5, ACM, Novi-Sad, Serbia (2021). DOI: 10.1145/3459960.3459967.
8. M. Popovic, B. Kordic.: PSTM: Python software transactional memory. In: 2014 2nd Telecommunications Forum Telfor, pp 1106-1109, TELFOR, Belgrade, Serbia (2014). DOI: 10.1109/TELFOR.2014.7034600.