



Francisco Figueiredo Goytacaz Sant'Anna

**Safe System-Level Concurrency on
Resource-Constrained Nodes with Céu**

Tese de Doutorado

Thesis presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática, PUC-Rio as partial fulfillment of the requirements for the degree of Doutor em Informática

Advisor : Prof. Roberto Ierusalimsky
Co-Advisor: Prof. Noemi de La Roque Rodriguez

Rio de Janeiro
September 2013



Francisco Figueiredo Goytacaz Sant'Anna

**Safe System-Level Concurrency on
Resource-Constrained Nodes with Céu**

Thesis presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática do Centro Técnico Científico da PUC-Rio, as partial fulfillment of the requirements for the degree of Doutor em Informática. Approved by the following commission:

Prof. Roberto Ierusalimsky

Advisor

Departamento de Informática — PUC-Rio

Prof. Noemi de La Roque Rodriguez

Co-Advisor

Departamento de Informática — PUC-Rio

Prof. Renato Fontoura de Gusmão Cerqueira

Departamento de Informática — PUC-Rio

Prof. Markus Endler

Departamento de Informática — PUC-Rio

Prof. Silvana Rossetto

UFRJ

Prof. Roberto da Silva Bigonha

UFMG

Prof. José Eugenio Leal

Head of the Centro Técnico Científico — PUC-Rio

Rio de Janeiro, September 12, 2013

All rights reserved.

Francisco Figueiredo Goytacaz Sant'Anna

He completed his undergraduate studies in Computer Engineering at the the Pontifical Catholic University of Rio de Janeiro (PUC–Rio) in 2003. He received his Master degree in Computer Science from PUC–Rio in 2009.

Bibliographic data

Sant'Anna, Francisco Figueiredo Goytacaz

Safe System-Level Concurrency on Resource-Constrained Nodes with Céu / Francisco Figueiredo Goytacaz Sant'Anna; advisor: Roberto Ierusalimschy; co–advisor: Noemi de La Roque Rodriguez. — Rio de Janeiro : PUC–Rio, Departamento de Informática, 2013.

88f. : il.; 30 cm

1. Tese de Doutorado - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Bibliography included.

1. Computer Science – Thesis. 2. Concurrency. 3. Determinism. 4. Embedded Systems. 5. Esterel. 6. Reactivity. 7. Synchronous. 8. Wireless Sensor Networks. I. Ierusalimschy, Roberto. II. Rodriguez, Noemi de La Roque. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Title.

CDD: 004

Acknowledgments

I would like to thank Roberto and Noemi for giving me freedom to make hits and misses for the last six years. Also, Phillippas and Olaf for kindly receiving me in Chalmers, Sweden. Luiz Fernando and Cerqueira for my period in the Telemídia Lab, where I started as a researcher. Finally, all my teachers in CAp-UERJ, responsible for my education before college.

My studies were funded by CNPq and SAAB.

Abstract

Sant’Anna, Francisco Figueiredo Goytacaz; Ierusalimschy, Roberto; Rodriguez, Noemi de La Roque. **Safe System-Level Concurrency on Resource-Constrained Nodes with Céu**. Rio de Janeiro, 2013. 88p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Despite the continuous research to facilitate Wireless Sensor Networks development, most safety analysis and mitigation efforts in concurrency are still left to developers, who must manage synchronization and shared memory explicitly. We propose a system language that ensures safe concurrency by handling threats at compile time, rather than at runtime. The synchronous and static foundation of our design allows for a simple reasoning about concurrency that enables compile-time analysis resulting in deterministic and memory-safe programs. As a trade-off, our design imposes limitations on the language expressiveness, such as doing computationally-intensive operations and meeting hard real-time responsiveness. To show that the achieved expressiveness and responsiveness is sufficient for a wide range of WSN applications, we implement widespread network protocols and the CC2420 radio driver. The implementations show a reduction in source code size, with a penalty of memory increase below 10% in comparison to *nesC*. Overall, we ensure safety properties for programs relying on high-level control abstractions that also lead to concise and readable code.

Keywords

Concurrency. Determinism. Embedded Systems. Esterel. Reactivity. Synchronous. Wireless Sensor Networks.

Resumo

Sant'Anna, Francisco Figueiredo Goytacaz; Ierusalimschy, Roberto; Rodriguez, Noemi de La Roque. **Concorrência Segura em Nível de Sistema para Nós com Restrições de Recursos em Céu.** Rio de Janeiro, 2013. 88p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Apesar da pesquisa contínua para facilitar a programação de redes de sensores sem fio, a análise de perigos de concorrência ainda é de responsabilidade do programador, que deve tratar manualmente de questões como sincronização e memória compartilhada. Nós apresentamos uma linguagem de sistema que garante concorrência segura tratando ameaças em tempo de compilação. A fundamentação estática e síncrona da nossa abordagem permite um raciocínio mais simples sobre questões de concorrência, permitindo uma análise em tempo de compilação que garante programas determinísticos. Como contra-partida, nosso modelo impõe em termos da expressividade da linguagem, tais como para efetuar cálculos demorados, ou atender prazos estritos em tempo real. Nós implementamos diversos protocolos de rede conhecidos e o driver para o rádio CC2420 para mostrar que a expressividade e responsividade obtida com a linguagem é suficiente para uma gama considerável de aplicações para redes de sensores. As implementações mostram uma redução de tamanho de código, com um aumento de memória abaixo de 10% em comparação com nesC. O uso da linguagem proposta implica em diversas propriedades de segurança que se baseiam em abstrações de controle de alto nível, também resultando em código mais conciso e legível.

Palavras-chave

Concorrência. Determinismo. Sistemas Embarcados. Esterel.
Síncrono. Reativo. Redes de Sensores sem Fio.

Contents

I	Introduction	13
II	Overview of programming models	17
II.1	Asynchronous model	17
II.2	Synchronous model	19
II.3	Programming models in WSNs	21
III	The design of Céu	25
III.1	The execution model of Céu	25
III.2	Shared-memory concurrency	31
III.3	Integration with <i>C</i>	33
III.4	Local scopes and finalization	34
III.5	First-class timers	37
III.6	Internal events	38
III.7	Differences to Esterel	41
IV	Demo applications	43
IV.1	WSN ring	43
IV.2	Spaceship game	46
V	Evaluation	51
V.1	Code size	53
V.2	Memory usage	54
V.3	Responsiveness	55
V.4	Battery consumption	57
V.5	Discussion	58
VI	The semantics of Céu	61
VI.1	Abstract syntax	61
VI.2	Operational semantics	62
VI.3	Concrete language mapping	67
VII	The implementation of Céu	71
VII.1	Temporal analysis	72
VII.2	Memory layout	73
VII.3	Trail allocation	73
VII.4	The external <i>C</i> API	77
VIII	Related work	81

<i>Contents</i>	8
-----------------	---

IX Conclusion	83
---------------	-----------

List of Figures

II.1	Two blinking LEDs in OCCAM-PI, ChibiOS and CÉU. Each line of execution in parallel blinks a LED with a fixed (but different) frequency. (The LEDs are connected to I/O ports 11 and 12.) Every 3 seconds both LEDs should light on together. After a couple of minutes of execution, only the implementation in CÉU remains synchronized.	18
II.2	“Blinking LED” in nesC, Protothreads, and CÉU.	22
III.1	Syntax of CÉU.	26
III.2	A CÉU program to illustrate the scheduler behavior.	27
III.3	A sequence of reaction chains for the program in Figure III.2.	28
III.4	Start/stop behavior for the radio driver. The occurrence of <code>CC2420_STOP</code> (line 5) seamlessly aborts the receiving loop (collapsed in line 9) and resets the driver to wait for the next <code>CC2420_START</code> (line 3).	30
III.5	Automatic detection for concurrent accesses to shared memory. The first example is suspicious because <code>x</code> and <code>p</code> can be accessed concurrently (lines 11 and 14). The second example is safe because accesses to <code>y</code> can only occur in sequence. The third example illustrates a false positive in our algorithm.	33
III.6	A CÉU program with embedded <i>C</i> definitions. The globals <code>I</code> and <code>inc</code> are defined in the <code>native</code> block (lines 3 and 4), and are imported by CÉU in line 8. <i>C</i> symbols must be prefixed with an underline to be used in CÉU (line 9).	34
III.7	Annotations for <i>C</i> functions. Function <code>abs</code> is side-effect free and can be concurrent with any other function. The functions <code>_Leds_led0Toggle</code> and <code>_Leds_led1Toggle</code> can execute concurrently. The variables <code>buf1</code> and <code>buf2</code> can be accessed concurrently (annotations are also applied to variables).	34
III.8	Unsafe use of local references. The period in which the radio driver manipulates the reference to <code>msg</code> passed by <code>_AMSend_send</code> (line 15) may outlive the lifetime of the variable scope, leading to an undefined behavior in the program.	36
III.9	A loop that awaits an internal event can emulate a subroutine. The <code>send</code> “subroutine” (lines 16-19) is invoked from three different parts of the program (lines 5, 9, and 14).	39
III.10	Exception handling in CÉU. The <code>emit</code> ’s in lines 10 and 14 raise an exception to be caught by the <code>await</code> in line 6. The <code>emit</code> continuations are discarded given that the surrounding <code>par/or</code> is aborted.	41
IV.1	Communicating trail for the WSN ring.	44
IV.2	Monitoring trail for the WSN ring.	45

IV.3	Retrying trail for the WSN ring.	46
IV.4	Retrying trail for the WSN ring.	47
IV.5	The “spaceship” game	47
IV.6	Outermost loop for the game.	48
IV.7	Sets the game attributes.	48
IV.8	The game central loop.	49
IV.9	The “game over” behavior for the game.	50
V.1	Comparison between CÉU and <i>nesC</i> for the implemented applications. The column group <i>Code size</i> compares the number of language tokens and global variables used in the sources; the group <i>Céu features</i> shows the number of times each functionality is used in each application; the group <i>Memory usage</i> compares ROM and RAM consumption.	52
V.2	Percentage of received packets depending on the duration of the lengthy operation. Note the logarithmic scale on the <i>x</i> -axis. The packet arrival frequency is 20ms. The operation frequency is 140ms. In the (left) green area, CÉU performs similarly to <i>nesC</i> . The (middle) gray area represents the region in which <i>nesC</i> is still responsive. In the (right) red area, both implementations become unresponsive (i.e. over 5% packet losses).	56
V.3	Percentage of received packets depending on the sending frequency. Each received packet is tied to a 8-ms operation. CÉU is 100% responsive up to a frequency of 30ms per packet.	57
V.4	Battery consumption for <i>nesC</i> and CÉU in the two experiments. The consumption line "Active" for the Experiment 1 is negligible, hence, the ratio between <i>nesC</i> and CÉU should not be considered.	58
VI.1	Reduced syntax of CÉU.	61
VI.2	The recursive predicate <i>isBlocked</i> is true only if all branches in parallel are hanged in <i>awaiting</i> or <i>emitting</i> expressions that cannot transit.	65
VI.3	The function <i>clear</i> extracts <i>fin</i> expressions in parallel and put their bodies in sequence.	66
VII.1	Compilation process: from the source code in CÉU to the final binary.	71
VII.2	A program with a corresponding AST describing the sets <i>I</i> and <i>O</i> . The program is safe because accesses to <i>y</i> in parallel have no intersections for <i>I</i> .	73
VII.3	A program with blocks in sequence and in parallel, with corresponding memory layout.	74
VII.4	Static allocation of trails and entry-point labels.	75
VII.5	Generated code for the program of Figure VII.4.	76
VII.6	The <i>TinyOS</i> binding for CÉU.	79

VIII.1 Table of features found in work related to CÉU.

The languages are sorted by the date they first appeared in a publication. A gray background indicates where the feature first appeared (or a contribution if it appears in a CÉU cell).

82

