

# Towards Software Verification for TinyOS Applications

Doina Bucur and Marta Kwiatkowska

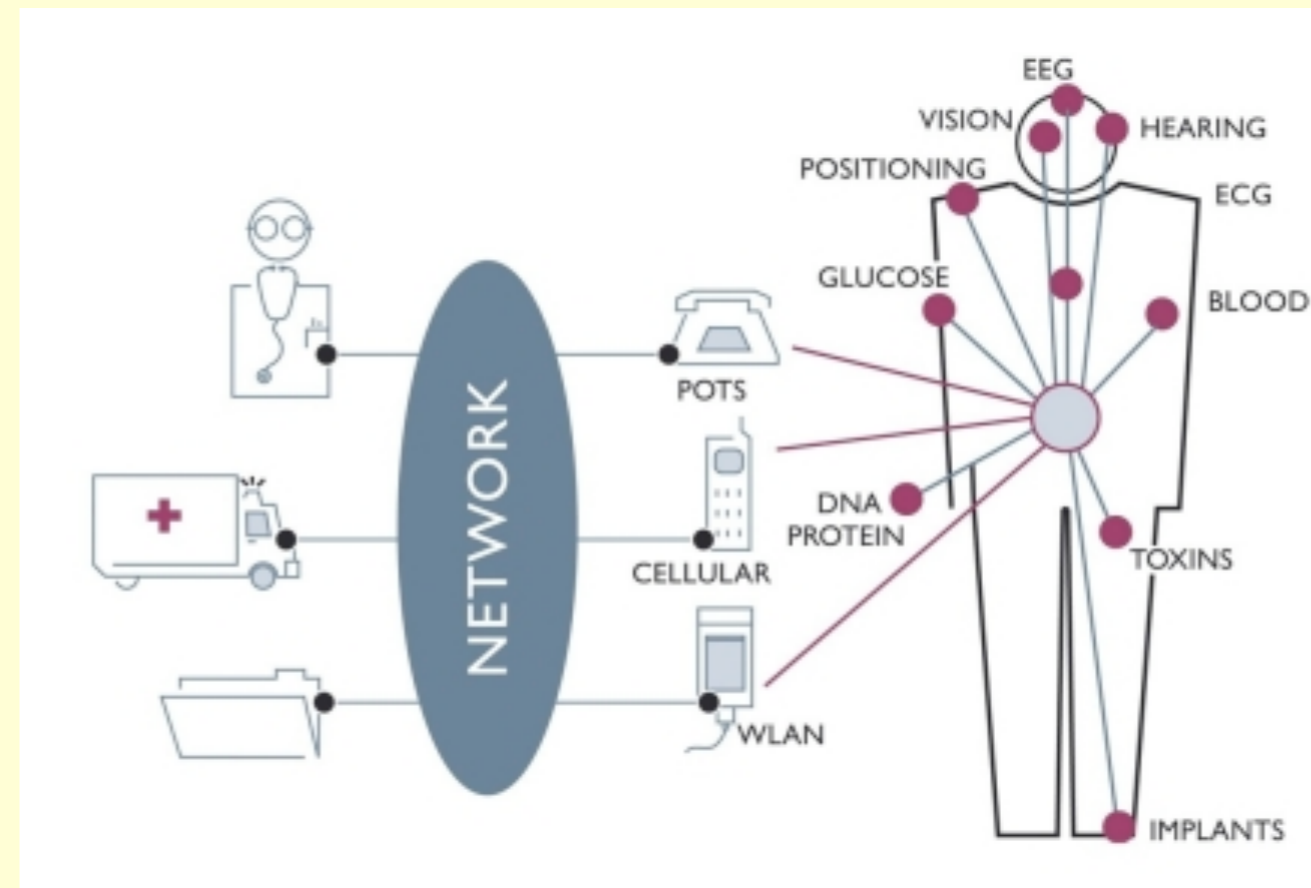


- Sensor networks
- Context-aware sensor software
- Context-aware safety specification
- Verification

# Overview

# Sensor networks

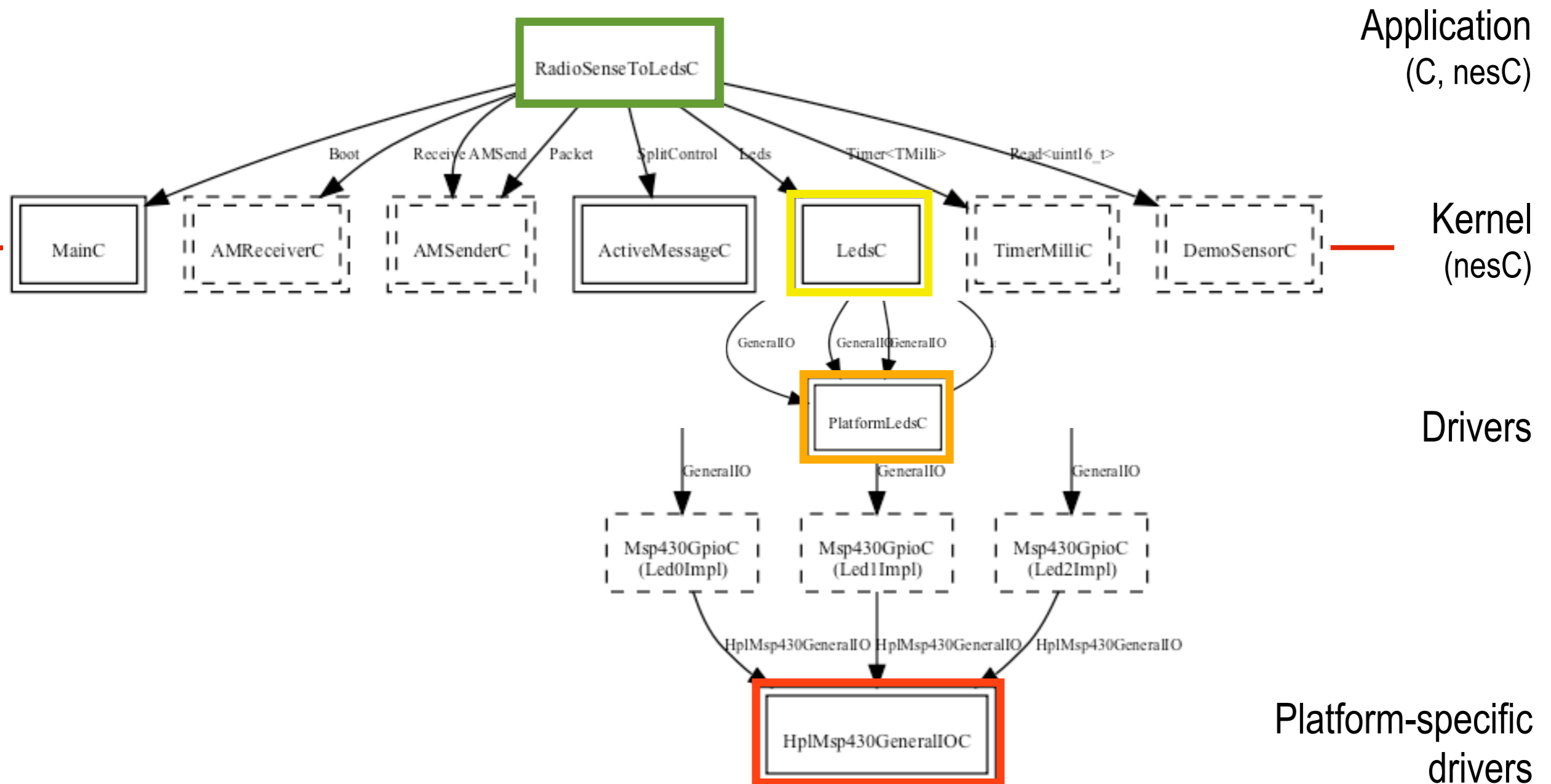
- Pervasive Healthcare.  
Body Sensor Networks  
[vip.doc.ic.ac.uk/bsn](http://vip.doc.ic.ac.uk/bsn)
- Sensor architecture / OS.  
TinyOS:
  - Modern OS and language in an embedded system



# Software verification

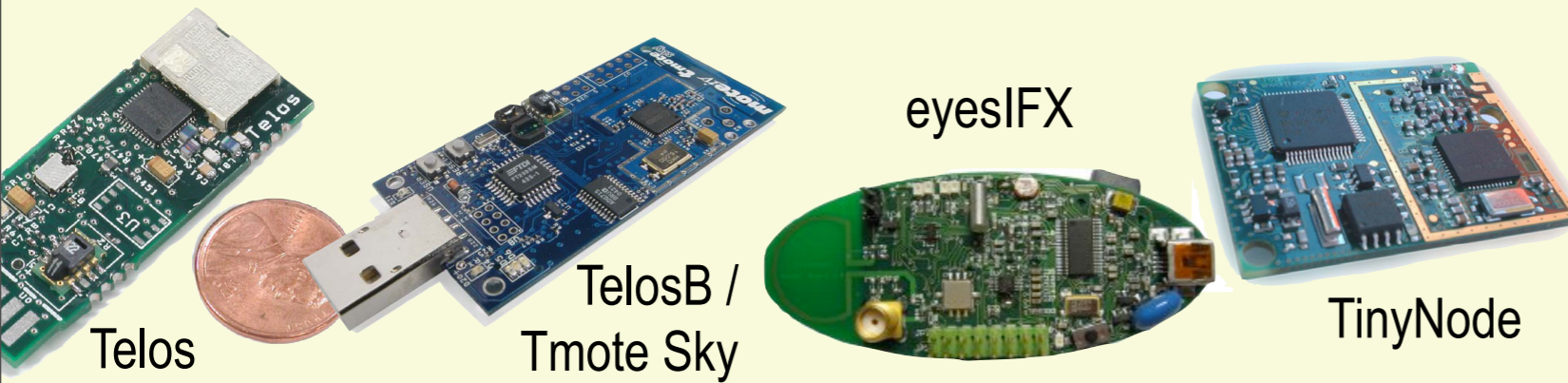
- **Verification** vs. simulation / formal verification
- Advances in software verification
- Counterexample-guided abstraction refinement (**CEGAR**)
- **CProver** tools [[cprover.org](http://cprover.org)].

# A TinyOS application

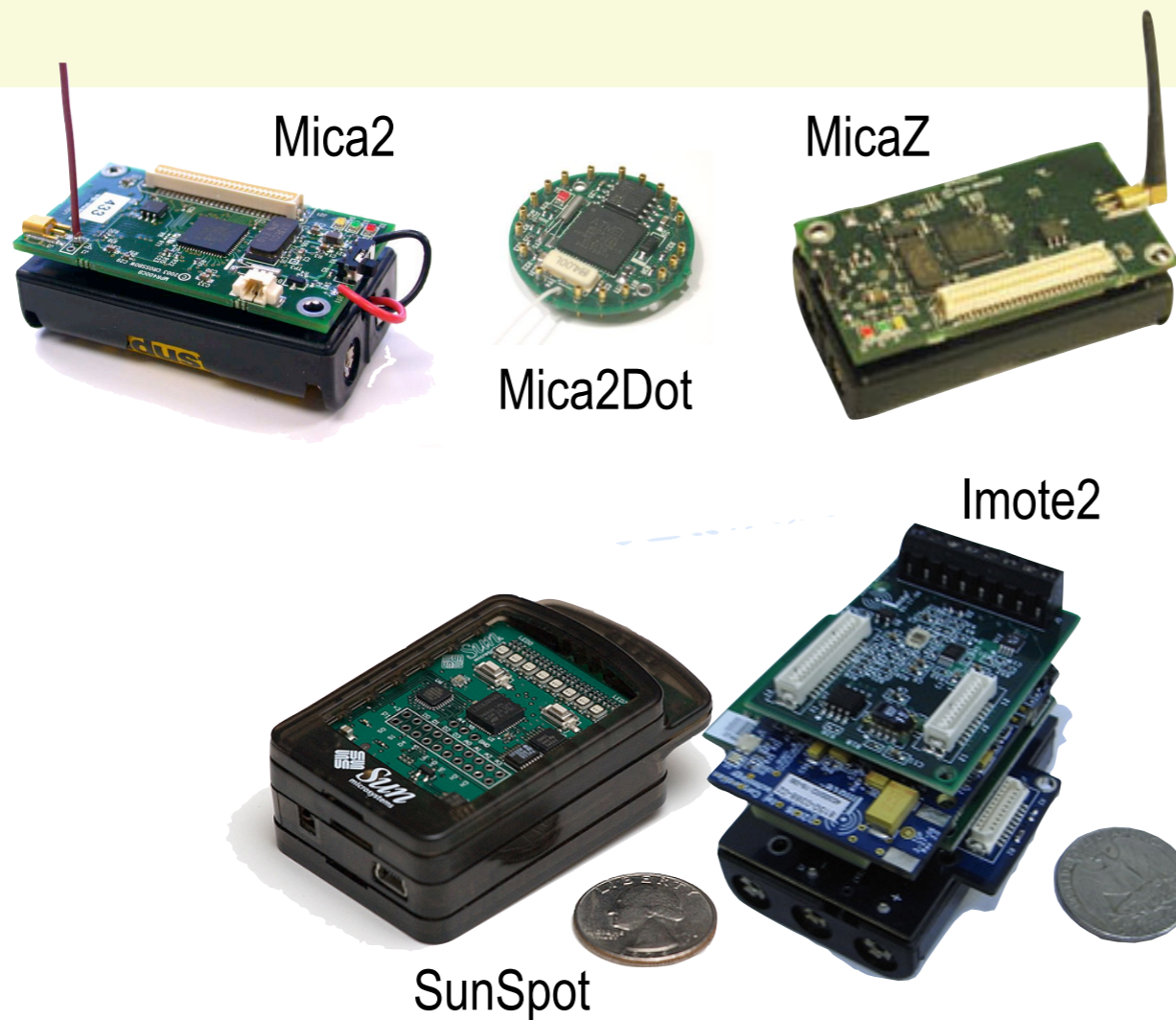


# Architectures

# CPU



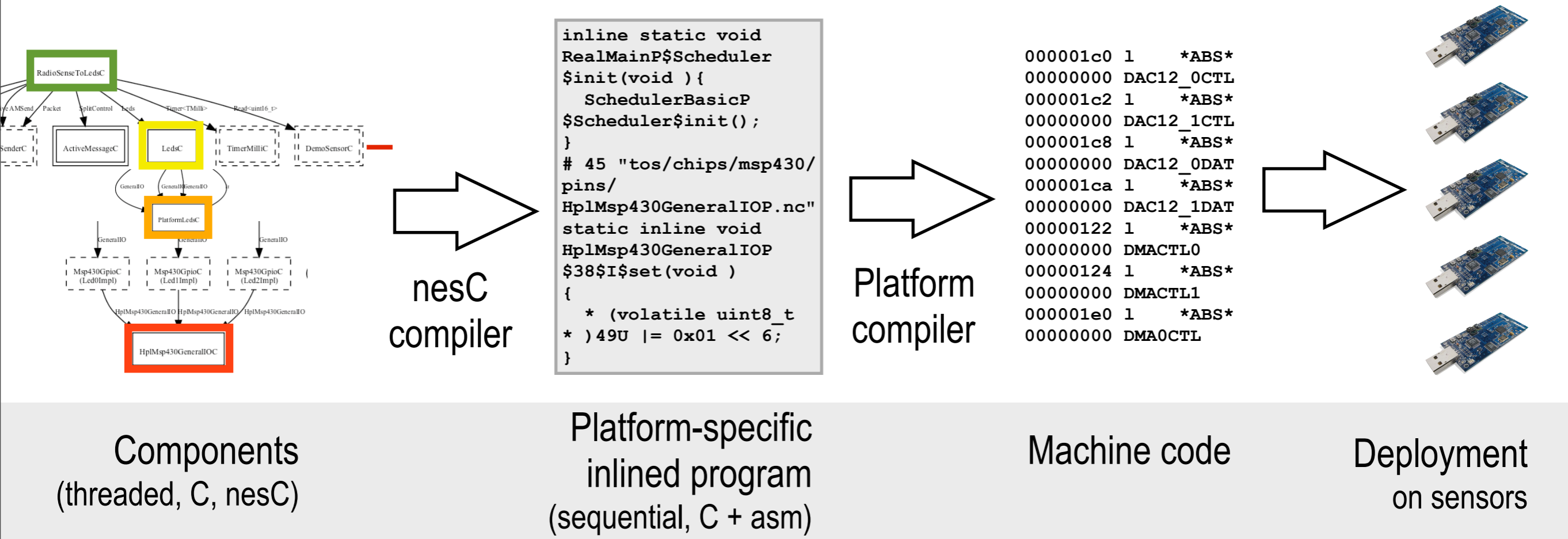
MSP430  
(Texas Instruments)



AVR  
(Atmel)

ARM

# ...and the software tool chain



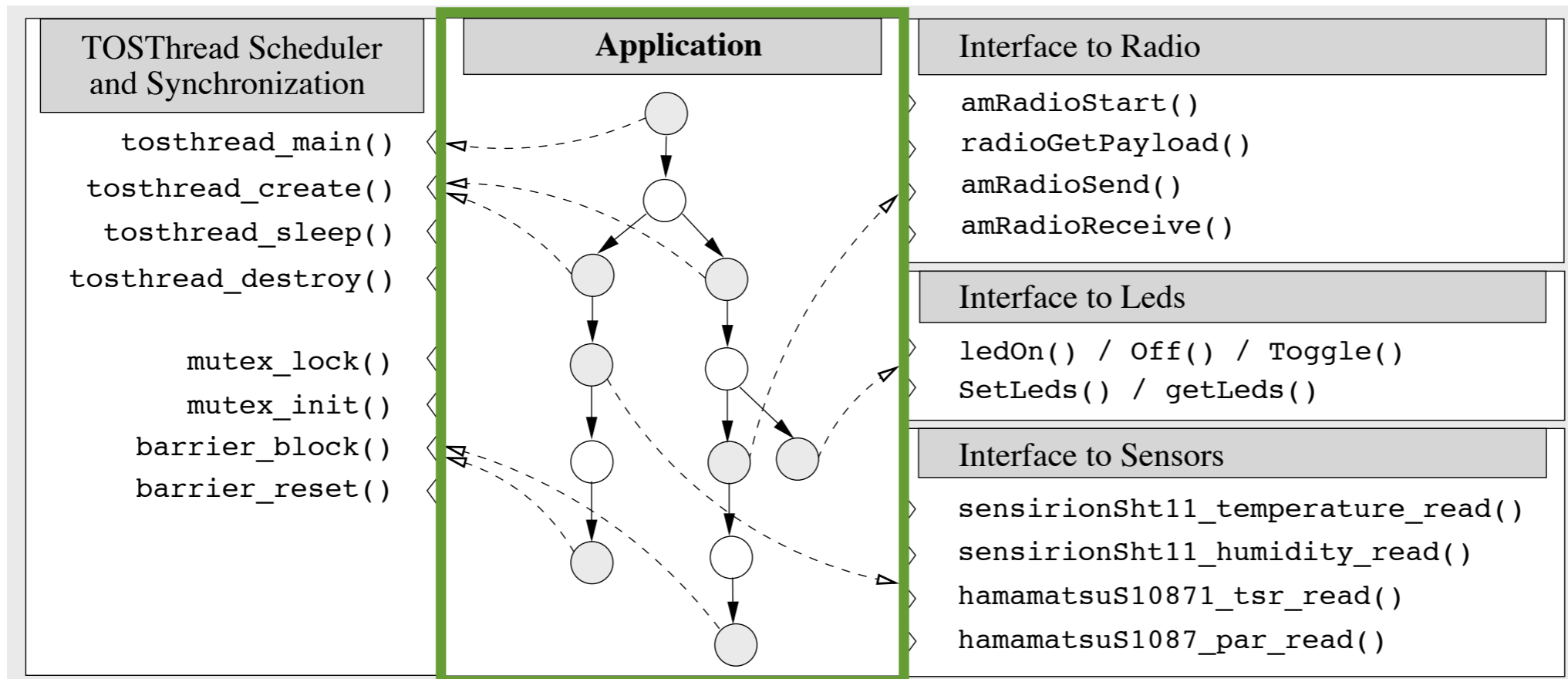
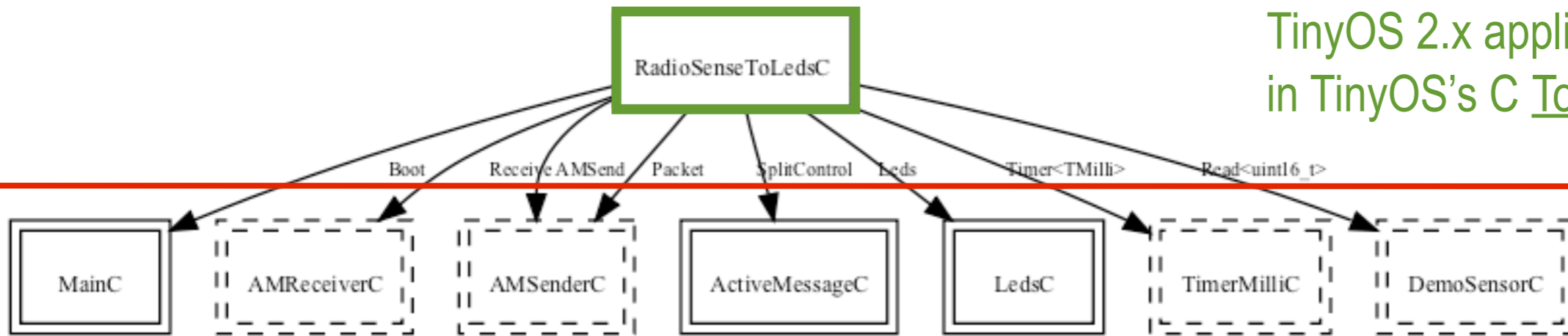
# Bug-Free Sensors: The Automatic Verification of Context-Aware TinyOS Applications

with **Marta Kwiatkowska**

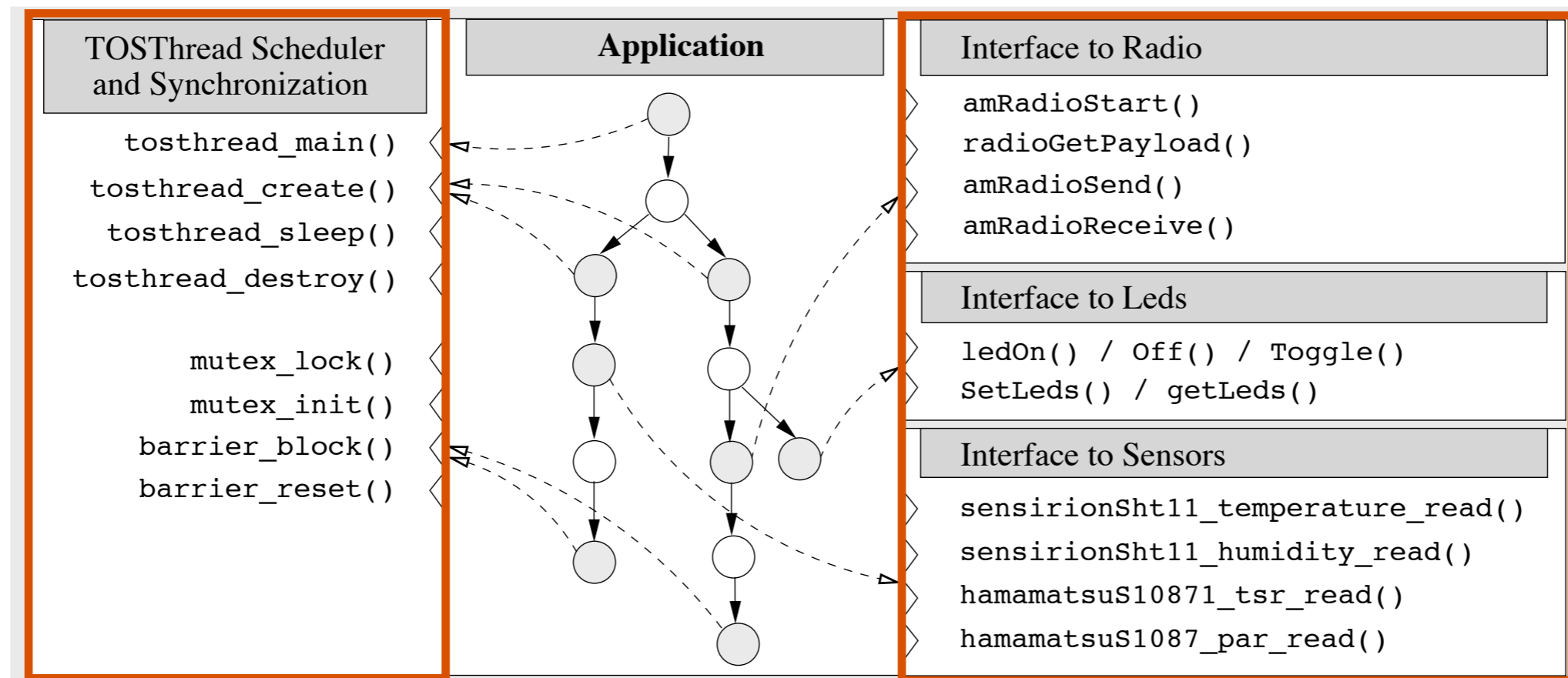
Proceedings of the **European Conference on Ambient Intelligence (Aml 2009)**  
Springer Lecture Notes in Computer Science (LNCS), Nov 2009

# Application model

TinyOS 2.x applications written in TinyOS's C TosThreads API.



# Kernel model



Model kernel services, ensuring that their interface behaviour is preserved.

# Programming errors and benchmarks

**Table 1.** Categories of bugs in context-aware, TinyOS applications

<b>Sensing exceptions</b>	Incomplete treatment of sensing errors.
<b>Network exceptions</b>	Incomplete treatment of network errors.
<b>Interface use</b>	Incorrect use of interface to kernel services.
<b>False reasoning</b>	Incorrect decision-making given a context situation.

**Table 2.** Categories of bugs in generic concurrent software

<b>Data race</b>	Multithreaded (write) access to shared resource. Not necessarily a bug.
<b>Atomicity violation</b>	Failure to enforce the atomicity of a code region.
<b>Order violation</b>	Failure to enforce execution order between two code regions.
<b>Deadlock</b>	A thread's failure to release a lock-like resource, halting execution.

Application (Threads/LOC)	Claim	Verified?	Time	Bug: context awareness	Bug: concurrency
<i>Blink</i> 4/64	66	yes	2.9s	-	-
<i>SenseAndSend</i> 6/347	79	no	32.2s	interface use	order violation
	136	no	1m08s	sensing exception	-
	146	yes	4m25s	-	-
<i>PatientNode</i> 6/439	172	yes	29.9s	(interface use)	(order violation)
	254	yes	3m55s	(sensing exception)	-
	230	no	35m07s	network exception	deadlock
	268	yes	2m38s	(false reasoning)	-
	262	yes	61m12s	(false reasoning)	-

# Rely-Guarantee Reasoning for Context-Aware Software

with **Marta Kwiatkowska**

In preparation.

# New safety specifications

- CProver specification style: **assertions**

```
sense_power (&ctx)           // sensing
...
if (power >= MIN_PWR) Led0On (); // actuation
...
259: assert (__red_led_on);
```

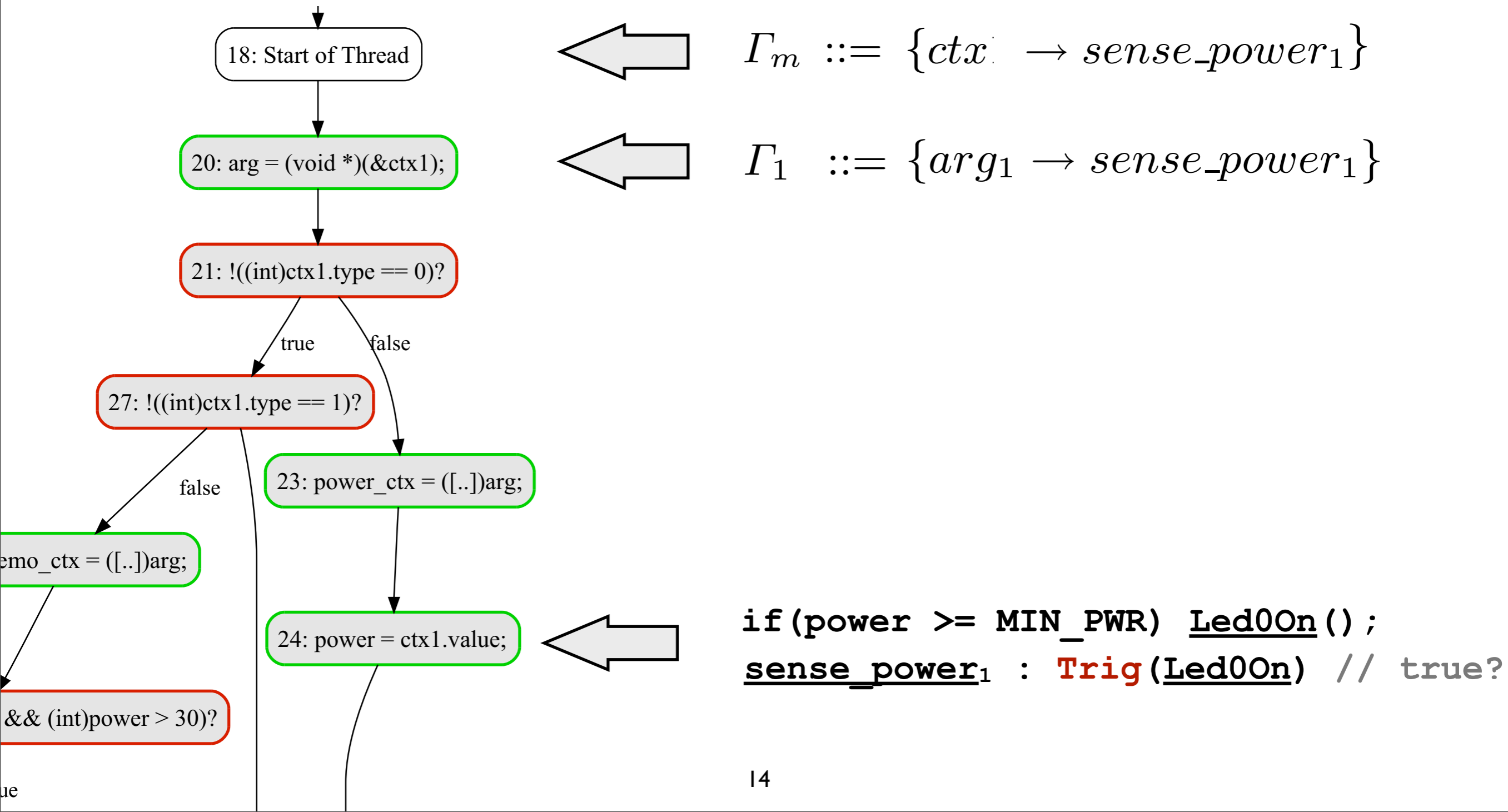
- We want **context-aware** assertions:

```
sense_power : Trig (Led0On)
```

... and thread-scalable verification.

# A typing system

## Side-effect and escape analysis



# A Rely-Guarantee logic --

- over “sensor” language:

(STMT)  $S ::= \underline{\text{sense}(\&x); \mid \text{actuate}(x); \mid \dots}$

- with thread-local rely/guarantee actions:

(ACTION)  $R, G ::= [p] \mid p \times q \mid \exists X.R$   
 $\mid R \vee R \mid R \wedge R \mid R \Rightarrow R \mid R * R$

- ... and assertions:

(ASSERTION)  $p, q, r ::= \text{emp}_s \mid \text{Own}(x) \mid B \mid \exists X.p$   
 $\mid \text{emp}_\alpha \mid \text{Own}(a) \mid \mathcal{P}(\text{Contexts}) : \text{Trig}(a) \mid a \mapsto n$   
 $\mid p \vee q \mid p \wedge q \mid p \Rightarrow q \mid p * q$

# A thread's verification

$\Gamma_m \vee \Gamma_1 \vee \Gamma_3 * \Gamma_2; R_2, G_2 \vdash$	$p_c \wedge p_4$ $\langle arg_2 := ctx2; \rangle$
$\Gamma_m \vee \Gamma_1 \vee \Gamma_3 * \Gamma_2; R_2, G_2 \vdash$	$p_{21} ::= p_c \wedge (arg_2 = Y)$ $\langle \text{IF } !(power > \text{MIN}) \text{ EXIT} \rangle$
$\Gamma_m \vee \Gamma_1 \vee \Gamma_3 * \Gamma'_2; R_2, G_2 \vdash$	$p_{22} ::= p_{21} \wedge (M > \text{MIN})$ $\langle out := DB \text{ sel } arg_2; \rangle$
$\Gamma_m \vee \Gamma_1 \vee \Gamma_3 * \Gamma'_2; R_2, G_2 \vdash$	$p_{23} ::= p_{22} \wedge (out = DB \text{ sel } Y)$ $\langle \text{IF } (power > \text{MIN}) \text{ display}(out); \rangle$
	$p_{23} \wedge q$

# Thank You!

## Software Verification for TinyOS Applications

Doina Bucur and Marta Kwiatkowska

