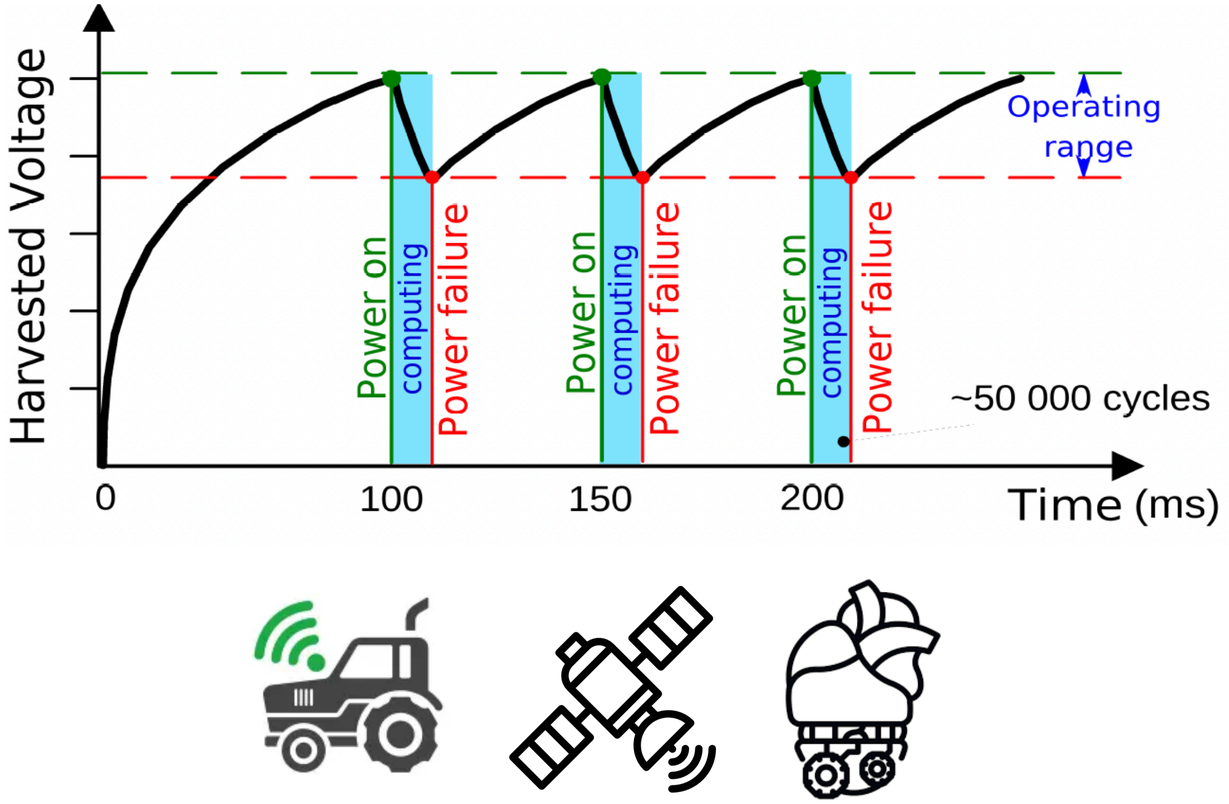
Correctness of Intermittent Executions via Modal Crash Types

Myra Dotzel
PhD student, CSD, CMU

Joint work with Farzaneh Derakhshan, Milijana Surbatovich, and Limin Jia

Intermittent Computing

- batteryless energy-harvesting devices deployed in harsh or inaccessible environments
- devices charge, compute, and crash frequently
- nonvolatile memory (NV) persistent while volatile memory (V) transient



Re-execution causes memory inconsistency bugs!

```
        Ckpt
L1      x := y;
L2      y := z;
L3      w := x + y
```

	x	y	z	w
	0	1	2	0
L1	1	1	2	0
L2	1	2	2	0
	1	2	2	0
L1	2	2	2	0
L2	2	2	2	0
L3	2	2	2	4

	x	y	z	w
	0	1	2	0
L1	1	1	2	0
L2	1	2	2	0
L3	1	2	2	3

↬ Write-after-read patterns [Lucia 2015]

Incorrect checkpointing causes bugs!

Question:

How can we **ensure** that intermittent programs execute correctly?

Surbatovich et al. 2019, 2020: practical implementation + correctness of intermittent executions

Overview of our work

- Model key operations: checkpoint, crash, restore, re-execute.
  - One approach: checkpoint everything not read-only (Derakhshan et al. 2023)
  - More efficient: only checkpoint write-after-read variables
- First type system based on modal logic to characterize the correctness of programs under crashes.

Types allow us to automatically check that programs can execute **correctly intermittently**.

Outline

- Background — intermittent computing, WAR patterns
- Overview
- **Type system**
- Logical relation
- Metatheory
- Conclusion

Type System

store types
basic types

$$A := \texttt{int} \mid \texttt{bool}$$
$$T := \texttt{unit} \mid A$$

unstable types
stable types

$$\tau^u := T \mid \downarrow \tau^s \mid \tau^u \vee \tau^u \mid v_t$$
$$\tau^s := \uparrow \tau^u \mid \texttt{nat} \rightsquigarrow \tau^s$$

access qualifiers

$$q := \texttt{CK} \mid \texttt{RD} \mid \texttt{MtWt} \mid \texttt{Wtn}$$

Type System

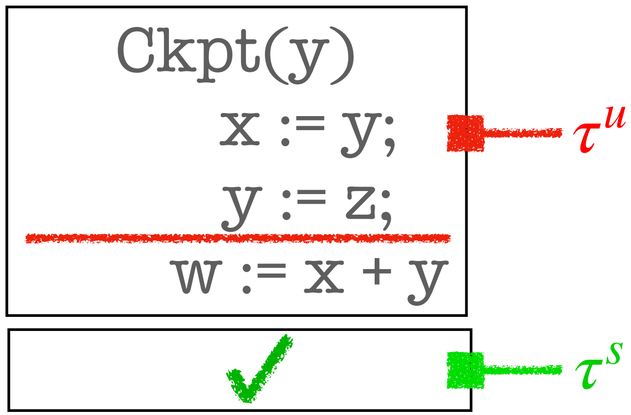
store types $A := \texttt{int} \mid \texttt{bool}$

basic types $T := \texttt{unit} \mid A$

unstable types $\tau^u := T \mid \downarrow \tau^s \mid \tau^u \vee \tau^u \mid v_t$

stable types $\tau^s := \uparrow \tau^u \mid \texttt{nat} \rightsquigarrow \tau^s$

access qualifiers $q := \texttt{CK} \mid \texttt{RD} \mid \texttt{MtWt} \mid \texttt{Wtn}$



Type System

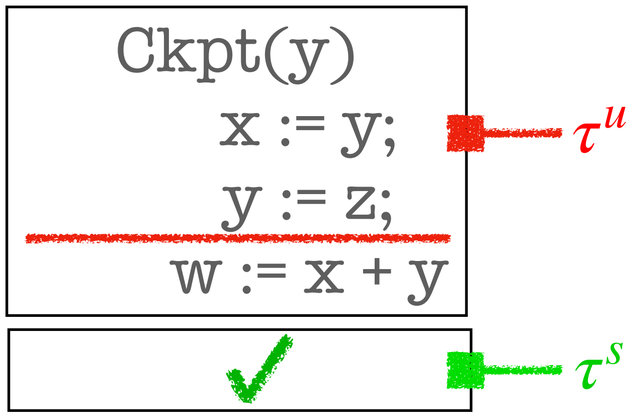
store types $A := \texttt{int} \mid \texttt{bool}$

basic types $T := \texttt{unit} \mid A$

unstable types $\tau^u := T \mid \downarrow \tau^s \mid \tau^u \vee \tau^u \mid v_t$

stable types $\tau^s := \uparrow \tau^u \mid \texttt{nat} \rightsquigarrow \tau^s$

access qualifiers $q := \texttt{CK} \mid \texttt{RD} \mid \texttt{MtWt} \mid \texttt{Wtn}$



Connection to Adjoint Modal Logic

- Independence principle: validity does not depend on truth.

A Valid

$\uparrow$ $\downarrow$

B True

- Here: stable values *(s)* do not depend on unstable values *(u)*.

$x : \tau^s$

Restore $\uparrow$ $\downarrow$ PwOff

$y : \tau^u$

Type System

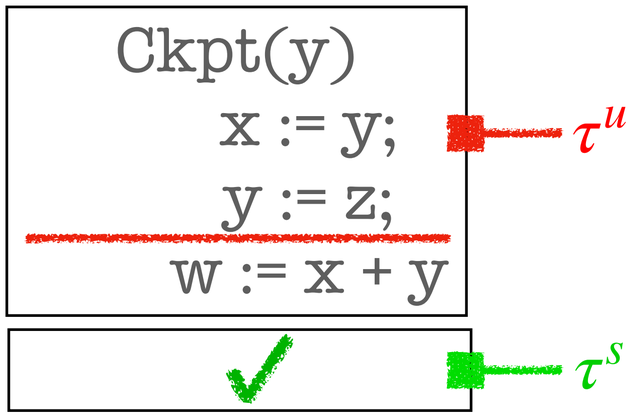
store types $A := \texttt{int} \mid \texttt{bool}$

basic types $T := \texttt{unit} \mid A$

unstable types $\tau^u := T \mid \downarrow \tau^s \mid \tau^u \vee \tau^u \mid v_t$

stable types $\tau^s := \uparrow \tau^u \mid \texttt{nat} \rightsquigarrow \tau^s$

access qualifiers $q := \texttt{CK} \mid \texttt{RD} \mid \texttt{MtWt} \mid \texttt{Wtn}$



Crash type

$$\texttt{C} = \downarrow \uparrow \texttt{unit} \vee \downarrow (\mathit{nat} \rightsquigarrow \uparrow \texttt{C})$$

Successful execution

Re-execution

Type System

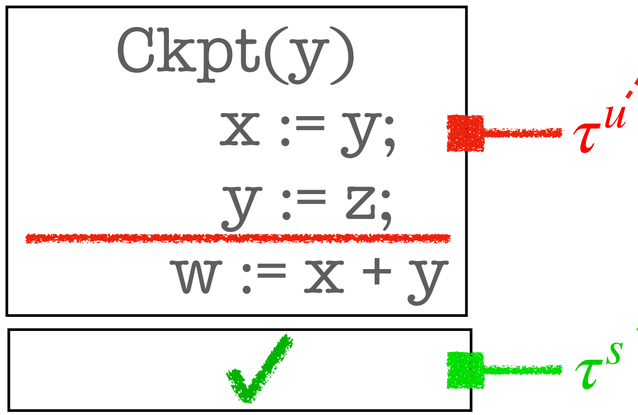
store types $A := \texttt{int} \mid \texttt{bool}$

basic types $T := \texttt{unit} \mid A$

unstable types $\tau^u := T \mid \downarrow \tau^s \mid \tau^u \vee \tau^u \mid v_t$

stable types $\tau^s := \uparrow \tau^u \mid \texttt{nat} \rightsquigarrow \tau^s$

access qualifiers $q := \texttt{CK} \mid \texttt{RD} \mid \texttt{MtWt} \mid \texttt{Wtn}$



Unstable typing judgment

$$x : \tau^s \quad y : \tau^u$$

$$\text{Energy buffer} \mid \Omega; \Sigma \vdash_{\texttt{Sig}} \texttt{x := y} : \texttt{C} \dashv \Omega'$$

Energy buffer | NV typing context | V typing context | Post-context

Stable typing judgment

$$x : \tau^s$$

$$\text{Energy buffer} \mid \Omega \vdash \texttt{Ckpt}\ \{\texttt{x := y; y := z; w := x + y}\} : \uparrow \texttt{C}$$

Type System

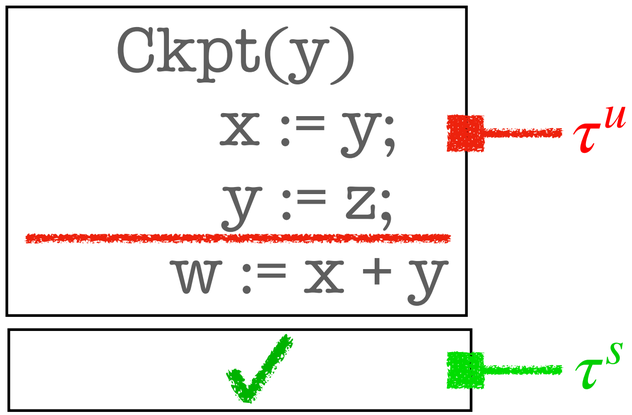
store types $A := \texttt{int} \mid \texttt{bool}$

basic types $T := \texttt{unit} \mid A$

unstable types $\tau^u := T \mid \downarrow \tau^s \mid \tau^u \vee \tau^u \mid v_t$

stable types $\tau^s := \uparrow \tau^u \mid \texttt{nat} \rightsquigarrow \tau^s$

access qualifiers $q := \texttt{CK} \mid \texttt{RD} \mid \texttt{MtWt} \mid \texttt{Wtn}$



Type System

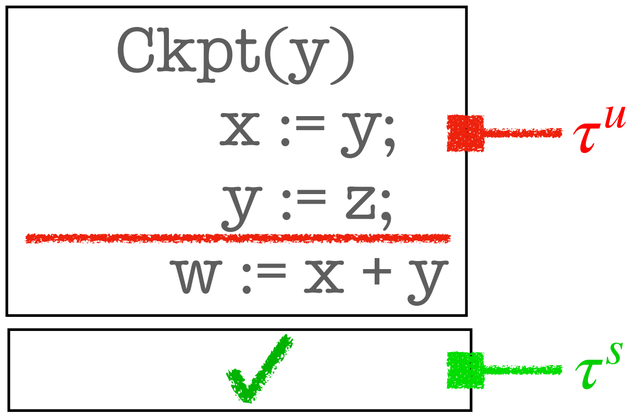
store types $A := \texttt{int} \mid \texttt{bool}$

basic types $T := \texttt{unit} \mid A$

unstable types $\tau^u := T \mid \downarrow \tau^s \mid \tau^u \vee \tau^u \mid v_t$

stable types $\tau^s := \uparrow \tau^u \mid \texttt{nat} \rightsquigarrow \tau^s$

access qualifiers $q := \texttt{CK} \mid \texttt{RD} \mid \texttt{MtWt} \mid \texttt{Wtn}$



Command sequencing

$$\frac{\blacksquare \mid \Omega; \Sigma \vdash_{\texttt{Sig}} \texttt{x} := \texttt{y} : \texttt{C}_{\texttt{unit}} \dashv \Omega' \qquad \blacksquare \mid \Omega'; \Sigma \vdash_{\texttt{Sig}} \texttt{y} := \texttt{z} : \tau \dashv \Omega''}{\blacksquare \mid \Omega; \Sigma \vdash_{\texttt{Sig}} \texttt{x} := \texttt{y}; \texttt{y} := \texttt{z} : \tau \dashv \Omega''}$$

$x{:}\tau\,@\,\texttt{MtWt}, y{:}\tau\,@\,\texttt{CK}$ $x{:}\tau\,@\,\texttt{Wtn}, y{:}\tau\,@\,\texttt{CK}$

$x{:}\tau\,@\,\texttt{Wtn}, y{:}\tau\,@\,\texttt{CK}$

Example

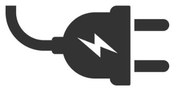
```
Ckpt(y)
L1      x := y;
L2      y := z;
L3      w := x + y
```

	x	y	z	w
	MtWt	CK	RD	MtWt
initial state	0	1	2	0
	Wtn	CK	RD	MtWt
L1	1	1	2	0
	Wtn	CK	RD	MtWt
L2	1	2	2	0

	x	y	z	w
	MtWt	CK	RD	MtWt
initial state	1	1	2	0
	Wtn	CK	RD	MtWt
L1	1	1	2	0
	Wtn	CK	RD	Wtn
L3	1	2	2	3

*

=



	x	y	z	w
	MtWt	CK	RD	MtWt
initial state	0	1	2	0
	Wtn	CK	RD	MtWt
L1	1	1	2	0
	Wtn	CK	RD	MtWt
L2	1	2	2	0
	Wtn	CK	RD	Wtn
L3	1	2	2	3

Outline

- Background — intermittent computing, WAR patterns
- Overview
- Type system
- **Logical relation**
- Metatheory
- Conclusion

Logical Relation

crashes

$$(\ \text{|NV | V |}\ c_1,\ \text{| NV| V |}\ c_2) \in \mathscr{E}[\![\mathrm{C}_{Unit}]\!]^m$$

Intermittent execution — Continuous execution — Crash type

Correctness:

Every intermittent execution of a well-typed program can be simulated by its continuous execution.

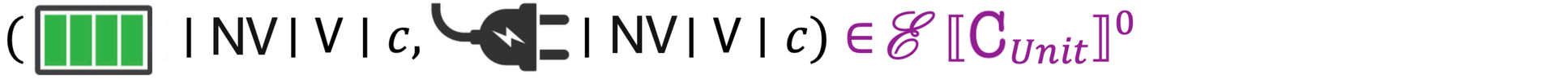
Term Relation

Base case

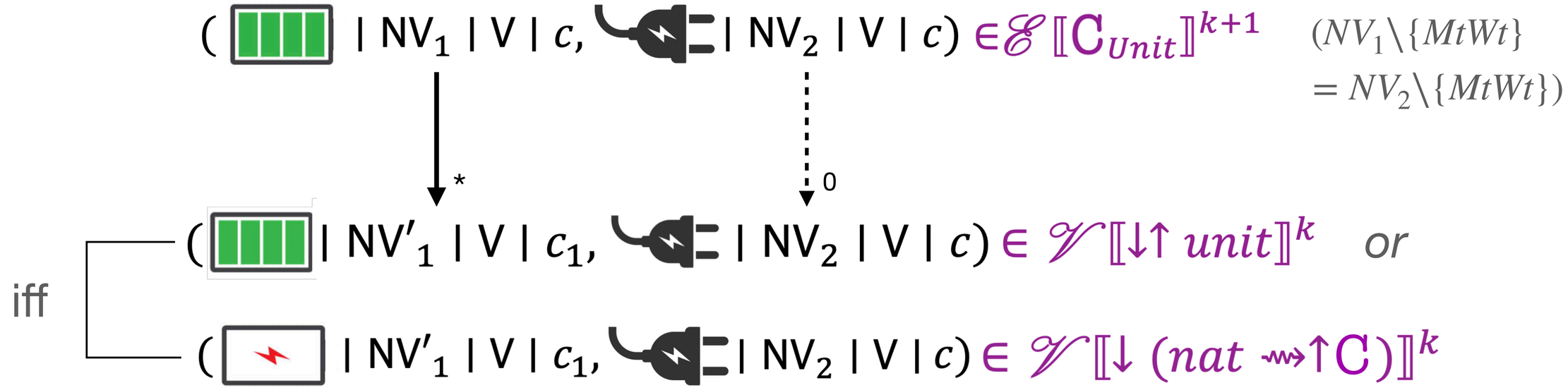
$$(\text{  } \mid \mathrm{NV} \mid \mathrm{V} \mid c, \text{  } \mid \mathrm{NV} \mid \mathrm{V} \mid c) \in \mathscr{E}[\![\mathbf{C}_{Unit}]\!]^0$$

Term Relation

Base case



Inductive case



$$C = \downarrow\uparrow \text{unit} \vee \downarrow(nat \rightsquigarrow \uparrow C)$$

Value Relation

Crash

Value Relation

$$(\boxed{\text{⚡}} \mid \mathsf{NV'_1} \mid \mathsf{V} \mid c_1, \text{🔌} \mid \mathsf{NV_2} \mid \mathsf{V} \mid c) \in \mathscr{V}[\![\mathbf{C}_{Unit}]\!]^{k+1} \quad \text{Crash}$$

iff $(\boxed{\text{⚡}} \mid \mathsf{NV'_1} \mid \mathsf{V} \mid \downarrow \epsilon\#in(b > 0, \uparrow c), \text{🔌} \mid \mathsf{NV_2} \mid \mathsf{V} \mid c) \in \mathscr{V}[\![\downarrow (nat \dashrightarrow \uparrow \mathbf{C}_{Unit})]\!]^{k}$ PwOff

Value Relation

$(\boxed{\text{⚡}} \mid \mathsf{NV}'_1 \mid \mathsf{V} \mid c_1, \text{🔌} \mid \mathsf{NV}_2 \mid \mathsf{V} \mid c) \in \mathscr{V}[\![\mathsf{C}_{Unit}]\!]^{k+1}$ Crash

iff $(\boxed{\text{⚡}} \mid \mathsf{NV}'_1 \mid \mathsf{V} \mid \downarrow \epsilon\#in(b > 0, \uparrow c), \text{🔌} \mid \mathsf{NV}_2 \mid \mathsf{V} \mid c) \in \mathscr{V}[\![\downarrow (nat \rightsquigarrow \uparrow \mathsf{C}_{Unit})]\!]^{k}$ PwOff

iff $(\boxed{\text{⚡}} \mid \mathsf{NV}'_1 \mid \epsilon\#in(b > 0, \uparrow c), \text{🔌} \mid \mathsf{NV}_2 \mid \mathsf{V} \mid c) \in \mathscr{V}[\![nat \rightsquigarrow \uparrow \mathsf{C}_{Unit}]\!]^{k}$ Harvest

Value Relation

$$(\boxed{\text{⚡}} \mid \mathsf{NV'}_1 \mid \mathsf{V} \mid c_1, \text{🔌} \mid \mathsf{NV}_2 \mid \mathsf{V} \mid c) \in \mathscr{V}[\![\mathsf{C}_{Unit}]\!]^{k+1} \quad \text{Crash}$$

$$\text{iff } (\boxed{\text{⚡}} \mid \mathsf{NV'}_1 \mid \mathsf{V} \mid \downarrow \epsilon\#in(b > 0, \uparrow c), \text{🔌} \mid \mathsf{NV}_2 \mid \mathsf{V} \mid c) \in \mathscr{V}[\![\downarrow (nat \rightsquigarrow \uparrow \mathsf{C}_{Unit})]\!]^{k} \quad \text{PwOff}$$

$$\text{iff } (\boxed{\text{⚡}} \mid \mathsf{NV'}_1 \mid \epsilon\#in(b > 0, \uparrow c), \text{🔌} \mid \mathsf{NV}_2 \mid \mathsf{V} \mid c) \in \mathscr{V}[\![nat \rightsquigarrow \uparrow \mathsf{C}_{Unit}]\!]^{k} \quad \text{Harvest}$$

$$\text{iff } (\boxed{\text{🔋}} \mid \mathsf{NV'}_1 \mid \uparrow c, \text{🔌} \mid \mathsf{NV}_2 \mid \mathsf{V} \mid c) \in \mathscr{V}[\![\uparrow \mathsf{C}_{Unit}]\!]^{k} \quad \text{Restore}$$

Value Relation

$(\boxed{\text{⚡}} \mid \mathsf{NV'}_1 \mid \mathsf{V} \mid c_1, \text{🔌} \mid \mathsf{NV}_2 \mid \mathsf{V} \mid c) \in \mathscr{V}[\![\mathsf{C}_{Unit}]\!]^{k+1}$ Crash

iff $(\boxed{\text{⚡}} \mid \mathsf{NV'}_1 \mid \mathsf{V} \mid \downarrow \epsilon\#in(b > 0, \uparrow c), \text{🔌} \mid \mathsf{NV}_2 \mid \mathsf{V} \mid c) \in \mathscr{V}[\![\downarrow (nat \rightsquigarrow \uparrow \mathsf{C}_{Unit})]\!]^{k}$ PwOff

iff $(\boxed{\text{⚡}} \mid \mathsf{NV'}_1 \mid \epsilon\#in(b > 0, \uparrow c), \text{🔌} \mid \mathsf{NV}_2 \mid \mathsf{V} \mid c) \in \mathscr{V}[\![nat \rightsquigarrow \uparrow \mathsf{C}_{Unit}]\!]^{k}$ Harvest

iff $(\boxed{\text{🔋}} \mid \mathsf{NV'}_1 \mid \uparrow c, \text{🔌} \mid \mathsf{NV}_2 \mid \mathsf{V} \mid c) \in \mathscr{V}[\![\uparrow \mathsf{C}_{Unit}]\!]^{k}$ Restore

iff $(\boxed{\text{🔋}} \mid \mathsf{NV''}_1 \mid \mathsf{V} \mid c, \text{🔌} \mid \mathsf{NV}_2 \mid \mathsf{V} \mid c) \in \mathscr{E}[\![\mathsf{C}_{Unit}]\!]^{k}$ $(NV''_1 \setminus \{MtWt\} = NV_2 \setminus \{MtWt\})$

Outline

- Background — intermittent computing, WAR patterns
- Overview
- Type system
- Logical relation
- **Metatheory**
- Conclusion

Metatheory

- Type Soundness: progress + preservation
- Fundamental Theorem: a syntactically well-typed program is semantically well-typed.
- Adequacy: semantically well-typed programs are correct.

Every intermittent execution of a well-typed program can be simulated by its continuous execution.

Outline

- Background — intermittent computing, WAR patterns
- Overview
- Type system
- Logical relation
- Metatheory
- **Conclusion**

Conclusion

- First logical interpretation of key operations of intermittent execution
- A core calculus for Crash types with type soundness
- A novel logical relation to characterize correctness
- Extensible to other modes of execution

Future work

- Other classes of bugs (repeated I/O)
- Shared memory concurrency
- Other systems crash too!

Correctness of Intermittent Executions via Modal Crash Types

Myra Dotzel
PhD student, CSD, CMU

Joint work with Farzaneh Derakhshan, Milijana Surbatovich, and Limin Jia

References

[1] I/O Dependent Idempotence Bugs in Intermittent Systems. Surbatovich et al. OOPSLA 2019.

[2] Towards a Formal Foundation of Intermittent Computing. Surbatovich et al. OOPSLA 2020.

[3] A simpler, safer programming and execution model for intermittent systems. Lucia et al. PLDI 2015.

[4] Modal Crash Types for Intermittent Computing. Derakhshan et al. ESOP 2023.

[5] A judgmental deconstruction of modal logic. Reed. 2009.

[6] A judgmental reconstruction of modal logic. Pfenning et al. IMLA'99.

[7] A mixed linear and non-linear logic: Proofs, terms and models. Benton. CLS'94.

[8] Adjoint Logic. Pruiksma et al. 2018.