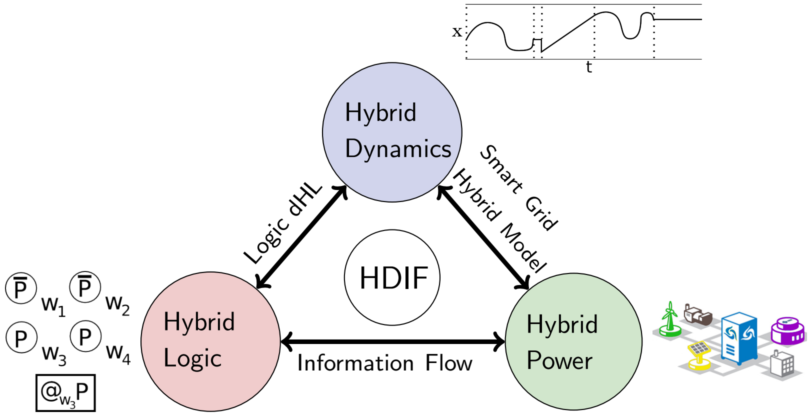
# A Hybrid, Dynamic Logic for Hybrid-Dynamic Information Flow

**Brandon Bohrer** and André Platzer

Logical Systems Lab
Computer Science Department
Carnegie Mellon University

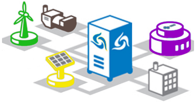
LICS'18

# Outline: Hybrid {Dynamics, Logic, Power}

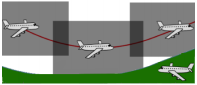



We 1) develop dHL, a hybrid *logic* for hybrid-*dynamical systems* and 2) apply dHL to verify hybrid dynamic information flow HDIF for 3) security of a hybrid *power grid*.

# CPS are Safety-Critical and Ubiquitous



Grid

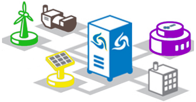


Transport


Medical

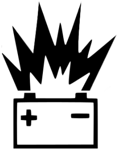
*How can we design cyber-physical systems people can bet their lives on? – Jeanette Wing*

# Secure Information Flow is Safety Critical

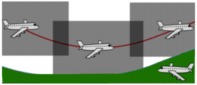


Grid

⇓



Overloads



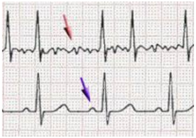
Transport

⇓

Position Spoofing

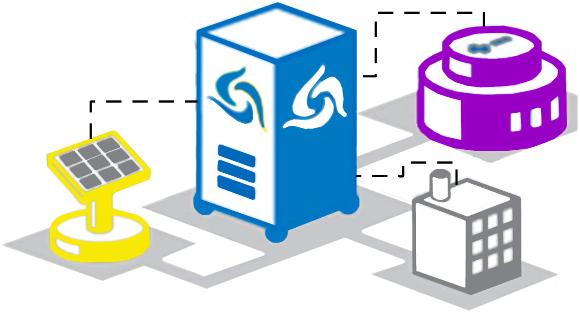

Medical

⇓



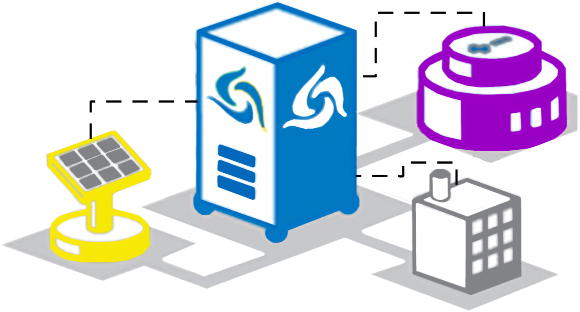
Hijacking

# Results Only as Good as the Model



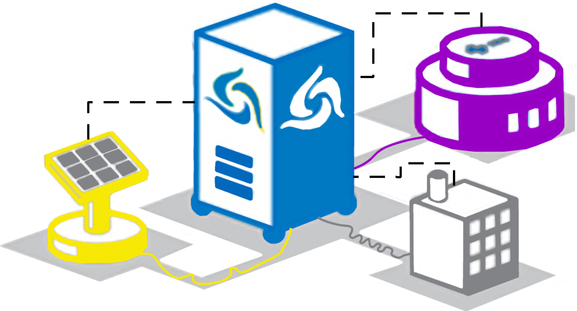
- Related work: Verified discrete event model of FREEDM grid
- Did not model physical dynamics

# Results Only as Good as the Model



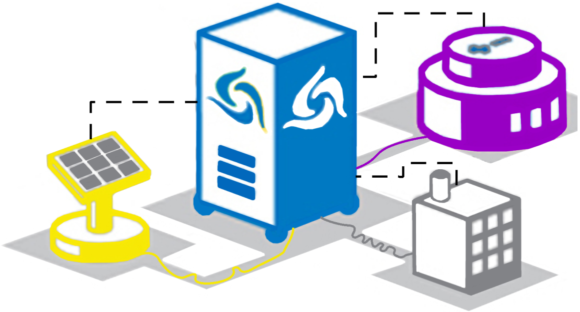
- Related work: Verified discrete event model of FREEDM grid
- Did not model physical dynamics
- **Event model can't catch vulnerabilities in dynamics!**

# Expressive Hybrid Models Provide Expressive Flows



- Hybrid dynamics: Mix and match discrete and continuous
- *Hybrid-Dynamic Information Flow* (HDIF): Information can flow in both **discrete and continuous channels**

# Expressive Hybrid Models Provide Expressive Flows



- Hybrid dynamics: Mix and match discrete and continuous
- *Hybrid-Dynamic Information Flow* (HDIF): Information can flow in both **discrete and continuous channels**
- **How do we model and verify HDIFs?**

# Outline

1. dHL: Hybrid {Dynamics, Logic}
2. FREEDM Case Study: Hybrid Power
3. Theory: Soundness and Reducibility

# Example Hybrid System: Diesel Generator

Generator consumes **Fuel** to produce **p**ower for the **gr**id.

$$\alpha_{\text{gen}} \overset{\text{def}}{\equiv} ((p := 0 \;\cup\; (p := *;\; ?(Fuel > 0 \wedge 0 \leq p \leq maxp));$$
$$\{Fuel' = -p,\; gr' = p \,\&\, Fuel \geq 0\})^*$$

**Questions:** Can grid observer detect fuel level?

| Program | Meaning |
|---|---|
| $\{x' = \theta \,\&\, \psi\}$ | Evolve ODE $x' = \theta$, but only while $\psi$ holds |
| $x := *$ | Assign randomly to $x$ |
| $?\phi$ | Test whether $\phi$ holds |
| $\alpha \cup \beta$ | Run $\alpha$ or $\beta$ |
| $\alpha^*$ | Run $\alpha$ any number of times in sequence |

# Dynamic Logic Operators

Definition (d$\mathcal{L}$ Formulas, Fragment of dH$\mathcal{L}$)

$$\phi, \psi ::= \phi \land \psi \mid \neg\phi \mid \exists x : \mathbb{R}\ \phi \mid \theta_1 \le \theta_2 \mid \langle\alpha\rangle\phi$$

- First-order classical logic
- Real-valued terms $\theta_1, \theta_2$
- Dynamic modality $\langle\alpha\rangle\phi$ says $\phi$ holds after some run of $\alpha$.

# Dynamic Logic Operators

$\alpha$ can reach $\phi$

Definition (d$\mathcal{L}$ Formulas, Fragment of dH$\mathcal{L}$)

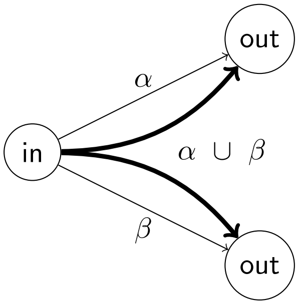
$$\phi, \psi ::= \phi \wedge \psi \mid \neg\phi \mid \exists x : \mathbb{R}\ \phi \mid \theta_1 \leq \theta_2 \mid \langle\alpha\rangle\phi$$

- First-order classical logic
- Real-valued terms $\theta_1, \theta_2$
- Dynamic modality $\langle\alpha\rangle\phi$ says $\phi$ holds after some run of $\alpha$.

# Program Axioms Decompose Dynamics

$\langle ' \rangle \quad \langle x' = F \,\&\, q(x) \rangle p(x) \leftrightarrow \exists t{\geq}0 (p(y(t)) \land \forall 0{\leq}s{\leq}t \; q(y(s)))$

$\langle \cup \rangle \quad \langle a \cup b \rangle P \leftrightarrow (\langle a \rangle P \lor \langle b \rangle P)$



# dH$\mathcal{L}$ Adds Hybrid Logic

Definition (dH$\mathcal{L}$, Hybrid-Logical Operators)

$$\phi ::= \cdots \mid @_w\phi \mid \exists s : \mathcal{W}\ \phi \mid \downarrow s\ \phi \mid w$$

- Evaluate formulas $\phi$ or terms $\theta$ and named world $w$.
- Quantifiers $\exists s : \mathcal{W}\ \phi, \forall s : \mathcal{W}\ \phi$, and $\downarrow s\ \phi$ (binds *current* world)
- Nominal predicate $w$ holds exactly in world named by $w$

@hom $\quad @_{\bar{n}} p(F_1, \ldots F_m) \leftrightarrow p(@_{\bar{n}} F_1, \ldots, @_{\bar{n}} F_m)$

$\downarrow \quad \downarrow s\ p(s) \leftrightarrow \exists s : \mathcal{W}\ (s \land p(s))$

@id $\quad @_{\bar{n}} \bar{n}$

Go to world $w$

# dH$\mathcal{L}$ Adds Hybrid Logic

Definition (dH$\mathcal{L}$, Hybrid-Logical Operators)

$$\phi ::= \cdots \mid @_w\phi \mid \exists s : \mathcal{W}\ \phi \mid \downarrow s\ \phi \mid w$$

- Evaluate formulas $\phi$ or terms $\theta$ and named world $w$.
- Quantifiers $\exists s : \mathcal{W}\ \phi, \forall s : \mathcal{W}\ \phi$, and $\downarrow s\ \phi$ (binds *current* world)
- Nominal predicate $w$ holds exactly in world named by $w$

| | |
|---|---|
| @hom | $@_{\bar{n}} p(F_1, \ldots F_m) \leftrightarrow p(@_{\bar{n}} F_1, \ldots, @_{\bar{n}} F_m)$ |
| $\downarrow$ | $\downarrow s\ p(s) \leftrightarrow \exists s : \mathcal{W}\ (s \wedge p(s))$ |
| @id | $@_{\bar{n}} \bar{n}$ |

Go to world $w$

Exists world

# $\mathcal{L}$ Adds Hybrid Logic

Definition (dH$\mathcal{L}$, Hybrid-Logical Operators)

$$\phi ::= \cdots \mid @_w\phi \mid \exists s : \mathcal{W}\ \phi \mid \downarrow s\ \phi \mid w$$

- Evaluate formulas $\phi$ or terms $\theta$ and named world $w$.
- Quantifiers $\exists s : \mathcal{W}\ \phi, \forall s : \mathcal{W}\ \phi$, and $\downarrow s\ \phi$ (binds *current* world)
- Nominal predicate $w$ holds exactly in world named by $w$

| | |
|---|---|
| @hom | $@_{\bar{n}}p(F_1, \ldots F_m) \leftrightarrow p(@_{\bar{n}}F_1, \ldots, @_{\bar{n}}F_m)$ |
| $\downarrow$ | $\downarrow s\ p(s) \leftrightarrow \exists s : \mathcal{W}\ (s \wedge p(s))$ |
| @id | $@_{\bar{n}}\bar{n}$ |

Go to world $w$

Exists world

Remember world in $s$

## $\mathcal{L}$ A … id Logic

### Definition (dH$\mathcal{L}$, Hybrid-Logical Operators)

$$\phi ::= \cdots \mid @_w\phi \mid \exists s : \mathcal{W}\ \phi \mid {\downarrow} s\ \phi \mid w$$

- Evaluate formulas $\phi$ or terms $\theta$ and named world $w$.
- Quantifiers $\exists s : \mathcal{W}\ \phi, \forall s : \mathcal{W}\ \phi$, and ${\downarrow} s\ \phi$ (binds *current* world)
- Nominal predicate $w$ holds exactly in world named by $w$

@hom $\quad @_{\bar{n}} p(F_1, \ldots F_m) \leftrightarrow p(@_{\bar{n}} F_1, \ldots, @_{\bar{n}} F_m)$

$\downarrow \quad {\downarrow} s\ p(s) \leftrightarrow \exists s : \mathcal{W}\ (s \wedge p(s))$

@id $\quad @_{\bar{n}} \bar{n}$

Go to world $w$ | Exists world | Remember world in $s$ | Test world

$\mathcal{L}$ A … id

**Definition (dH$\mathcal{L}$, Hybrid-Logical Operators)**

$$\phi ::= \cdots \mid @_w\phi \mid \exists s : \mathcal{W}\ \phi \mid {\downarrow} s\ \phi \mid w$$

- Evaluate formulas $\phi$ or terms $\theta$ and named world $w$.
- Quantifiers $\exists s : \mathcal{W}\ \phi, \forall s : \mathcal{W}\ \phi$, and ${\downarrow} s\ \phi$ (binds *current* world)
- Nominal predicate $w$ holds exactly in world named by $w$

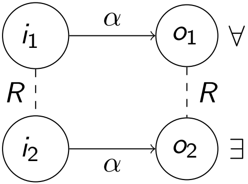
| | |
|---|---|
| @hom | $@_{\bar{n}}p(F_1, \ldots F_m) \leftrightarrow p(@_{\bar{n}}F_1, \ldots, @_{\bar{n}}F_m)$ |
| $\downarrow$ | ${\downarrow} s\ p(s) \leftrightarrow \exists s : \mathcal{W}\ (s \wedge p(s))$ |
| @id | $@_{\bar{n}}\bar{n}$ |

# Nondeducibility Information Flow

Program $\alpha$ is *nondeducibility*-secure with bisimulation $R$ when

$$\forall i_1, i_2, o_1 : \mathcal{W} \ \big(@_{i_1}\langle\alpha\rangle o_1 \wedge R(i_1, i_2) \rightarrow \ @_{i_2}\langle\alpha\rangle{\downarrow}o_2 \ R(o_1, o_2)\big)$$

$$R(k_1, k_2) \stackrel{\text{def}}{\equiv} \bigwedge_{\theta \in L} (@_{k_1}\theta = @_{k_2}\theta) \quad \text{(i.e., } k_1, k_2 \text{ agree on } L\text{)}$$

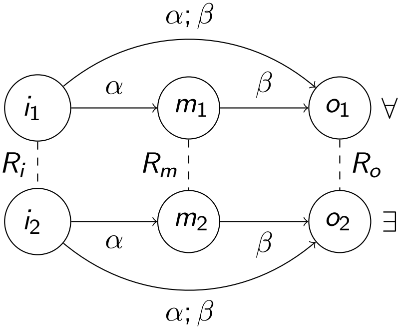



"All similar inputs would have made similar outputs possible"

# Derived Rules Simplify HDIF Proofs

Relational reasoning proceeds structurally on programs

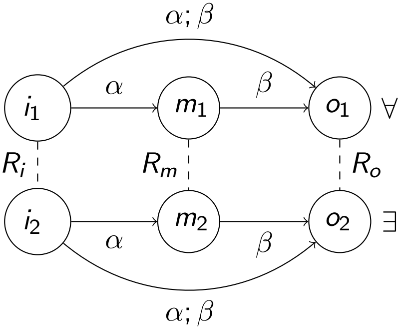
$$\text{BS;}\quad \frac{@_{i_1}\langle\alpha\rangle m_1 \wedge R_i(i_1, i_2) \rightarrow @_{i_2}\langle\alpha\rangle{\downarrow}m_2\ R_m(m_1, m_2) \qquad @_{m_1}\langle\beta\rangle o_1 \wedge R_m(m_1, m_2) \rightarrow @_{m_2}\langle\beta\rangle{\downarrow}o_2\ R_o(o_1, o_2)}{@_{i_1}\langle\alpha;\beta\rangle o_1 \wedge R_i(i_1, i_2) \rightarrow @_{i_2}\langle\alpha;\beta\rangle{\downarrow}o_2\ R_o(o_1, o_2)}$$



# Derived Rules Simplify HDIF Proofs

Relational reasoning proceeds structurally on programs

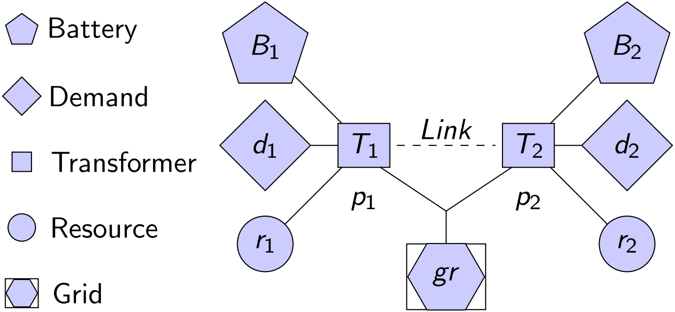
$$\text{BS;}\quad \frac{@_{i_1}\langle\alpha\rangle m_1 \wedge R_i(i_1, i_2) \rightarrow @_{i_2}\langle\alpha\rangle{\downarrow}m_2\ R_m(m_1, m_2) \qquad @_{m_1}\langle\beta\rangle o_1 \wedge R_m(m_1, m_2) \rightarrow @_{m_2}\langle\beta\rangle{\downarrow}o_2\ R_o(o_1, o_2)}{@_{i_1}\langle\alpha;\beta\rangle o_1 \wedge R_i(i_1, i_2) \rightarrow @_{i_2}\langle\alpha;\beta\rangle{\downarrow}o_2\ R_o(o_1, o_2)}$$




Bisimulation rules are all **derived!**

# Outline

# Example: FREEDM Smart Grid




Our hybrid model reveals a bug missed by the event-based model

# FREEDM: Formal Model

$$\alpha_F \equiv (\text{ctrl}; \text{plant})^* \qquad \text{ctrl} \equiv \underline{\text{migrate}; \text{bat}}$$

$$\begin{aligned} \text{migrate} \equiv \{\ & d_i, r_i := *; ?(d_i, r_i \geq 0);\ n_i := d_i - (r_i + p_i); \\ & \texttt{if}\ (n_i \geq \mathit{thresh} \wedge n_{\bar{i}} < 0) \quad \{\ m := \text{Migrate}(i)\} \\ & \texttt{else} \qquad\qquad\qquad\qquad\qquad\ \{\ m := 0\}\} \end{aligned}$$

$$\text{plant} \equiv \{p_i' = -1^i \cdot m, B_i' = b_i, b_i' = bm_i, gr' = grm, t' = 1\ \&\ B_i \geq 0\}$$

$\underline{\text{bat}_I} \equiv$

$gr, bm_i, vGridMig := 0;$

$\texttt{if}\ ((\mathbf{n_i} \leq \mathbf{0}\ \wedge\ \neg\text{Full}) \vee (n_i > 0 \wedge \neg\text{Emp}))\{$

$\quad \{\ \text{ToBat}(n_i, m)\}$

$\texttt{else}\ \{\ \text{ToGrid}(n_i, m)\}$

$\underline{\text{bat}_S} \equiv$

$gr, bm_i, vGridMig := 0;$

$\big(\mathbf{?}(\text{Full} \vee (n_i > 0 \wedge \neg\text{Emp}));$

$\quad \text{ToBat}(n_i, m)\big)$

$\cup\, (\text{ToGrid}(n_i, m))$

## FREEDM: Formal Model

Load Balance

$$\alpha_F \equiv (\text{ctrl}; \text{plant})^* \qquad \text{ctrl} \equiv \underline{\text{migrate}; \text{bat}}$$

$$\begin{aligned}\text{migrate} \equiv \{\ & d_i, r_i := *; ?(d_i, r_i \geq 0);\ \ n_i := d_i - (r_i + p_i);\\ & \texttt{if}\ (n_i \geq \mathit{thresh} \wedge n_{\bar{i}} < 0)\ \ \{\ m := \text{Migrate}(i)\}\\ & \texttt{else}\ \qquad\qquad\qquad\qquad \{\ m := 0\}\}\end{aligned}$$

$$\text{plant} \equiv \{p_i' = -1^i \cdot m, B_i' = b_i, b_i' = bm_i, gr' = grm, t' = 1\ \&\ B_i \geq 0\}$$

$\underline{\text{bat}}_I \equiv$

$gr, bm_i, vGridMig := 0;$

$\texttt{if}\ ((\mathbf{n_i} \leq \mathbf{0}\ \wedge\ \neg\text{Full}) \vee (n_i > 0 \wedge \neg\text{Emp}))\{$

$\quad \{\ \text{ToBat}(n_i, m)\}$

$\texttt{else}\ \{\ \text{ToGrid}(n_i, m)\}$

$\underline{\text{bat}}_S \equiv$

$gr, bm_i, vGridMig := 0;$

$\big(\mathbf{?}(\text{Full} \vee (n_i > 0 \wedge \neg\text{Emp}));$

$\quad \text{ToBat}(n_i, m)\big)$

$\cup\, (\text{ToGrid}(n_i, m))$

# FREEDM: Formal Model

Load Balance

$\alpha_F \equiv (\text{ctrl}; \text{plant})^* \qquad \text{ctrl} \equiv \underline{\text{migrate}; \text{bat}}$

$$\text{migrate} \equiv \{\ d_i, r_i := *; ?(d_i, r_i \geq 0);\ n_i := d_i - (r_i + p_i);$$
$$\texttt{if}\ (n_i \geq \mathit{thresh} \land n_{\bar{i}} < 0)\ \{\ m := \text{Migrate}(i)\}$$
$$\texttt{else}\ \{\ m := 0\}\}$$

Battery, Insecure

$$\equiv \{p_i' = -1^i \cdot m, B_i' = b_i, b_i' = bm_i, gr' = grm, t' = 1\ \&\ B_i \geq 0\}$$

$\underline{\text{bat}_I} \equiv$

$gr, bm_i, vGridMig := 0;$

$\texttt{if}\ ((\mathbf{n_i} \leq \mathbf{0}\ \land\ \neg\text{Full}) \lor (n_i > 0 \land \neg\text{Emp}))\{$

$\quad \{\ \text{ToBat}(n_i, m)\}$

$\texttt{else}\ \{\ \text{ToGrid}(n_i, m)\}$

$\underline{\text{bat}_S} \equiv$

$gr, bm_i, vGridMig := 0;$

$\big(\mathbf{?}(\text{Full} \lor (n_i > 0 \land \neg\text{Emp}));$

$\quad \text{ToBat}(n_i, m)\big)$

$\cup\,(\text{ToGrid}(n_i, m))$

# FREEDM: Formal Model

Load Balance

$\alpha_F \equiv (\text{ctrl}; \text{plant})^* \qquad \text{ctrl} \equiv \underline{\text{migrate}; \text{bat}}$

$$\text{migrate} \equiv \{\ d_i, r_i := *; ?(d_i, r_i \geq 0);\ n_i := d_i - (r_i + p_i);$$
$$\texttt{if}\ (n_i \geq \mathit{thresh} \wedge n_{\bar{i}} < 0)\ \{\ m := \text{Migrate}(i)\}$$
$$\texttt{else}\ \{\ m := 0\}\}$$

Battery, Insecure

$$\equiv \{p_i' = -1^i \cdot m, B_i' = b_i, b_i' = bm_i, gr' = grm, t' = \ldots$$

Battery, Secure

$\underline{\text{bat}}_I \equiv$

$gr, bm_i, vGridMig := 0;$

$\texttt{if}\ ((\mathbf{n_i} \leq \mathbf{0}\ \wedge\ \neg\text{Full}) \vee (n_i > 0 \wedge \neg\text{Emp}))\{$

$\quad \{\ \text{ToBat}(n_i, m)\}$

$\texttt{else}\ \{\ \text{ToGrid}(n_i, m)\}$

$\underline{\text{bat}}_S \equiv$

$gr, bm_i, vGridMig := 0;$

$\big(\mathbf{?}(\text{Full} \vee (n_i > 0 \wedge \neg\text{Emp}));$

$\quad \text{ToBat}(n_i, m)\big)$

$\cup (\text{ToGrid}(n_i, m))$

# FREEDM: Results

Define $R(i, j) \equiv (@_i t = @_j t \wedge @_i gr = @_j gr)$.
Same grid flow, same time

Proposition (FREEDM with original $\text{bat}_I$ is insecure)

$\exists i_1, i_2, o_1 : \mathcal{W}\big(@_{i_1}\langle\alpha_I\rangle o_1 \wedge R(i_1, i_2) \wedge @_{i_2}[\alpha_I]{\downarrow}o_2\ \neg R(o_1, o_2)\big)$

Proposition (Nondeducibility for fixed FREEDM)

$\forall i_1, i_2, o_1 : \mathcal{W}\ \big(@_{i_1}\langle\alpha_S\rangle o_1 \wedge R(i_1, i_2) \rightarrow @_{i_2}\langle\alpha_S\rangle{\downarrow}o_2\ R(o_1, o_2)\big)$

**Takeaway:** Determinism helps attackers! ("Refinement Paradox")

# FREEDM: Results

Define $R(i,j) \equiv (@_i t = @_j t \wedge @_i gr = @_j gr)$.
Same grid flow, same time

Proposition (FREEDM with original $\text{bat}_I$ is insecure)
$\exists i_1, i_2, o_1 : \mathcal{W}\big(@_{i_1}\langle\alpha_I\rangle o_1 \wedge R(i_1,i_2) \wedge @_{i_2}[\alpha_I]{\downarrow}o_2\ \neg R(o_1,o_2)\big)$

Proposition (Nondeducibility for fixed FREEDM)
$\forall i_1, i_2, o_1 : \mathcal{W}\ \big(@_{i_1}\langle\alpha_S\rangle o_1 \wedge R(i_1,i_2) \rightarrow @_{i_2}\langle\alpha_S\rangle{\downarrow}o_2\ R(o_1,o_2)\big)$

**Takeaway:** Determinism helps attackers! ("Refinement Paradox")
**Impact:** Translates to, e.g., randomization in implementation.

# Outline

# Hybrid Logic (+Uniform Substitution)

## Provides Clean Foundation for Info. Flow

Ours is a *uniform substitution* calculus: variables over predicates, programs, etc. represented *explicitly* in concrete axiom formulas, instantiated with rule US:

$$\text{US} \quad \frac{\phi}{\sigma(\phi)}$$

Rule US sound iff $\sigma$ is *admissible:*

### Definition (Admissibility (d$\mathcal{L}$))

Substitution $\sigma$ adds no free **variable** references in bound positions

### Definition (Admissibility (dH$\mathcal{L}$))

Substitution $\sigma$ adds no free **symbol** references in bound positions

**Takeaway:** Admissibility generalizes cleanly to hybrid logics

# Axiom Validity

**Proposition (dHL contains d$\mathcal{L}$)**

*A* d$\mathcal{L}$ *formula* $\phi$ *is valid in* d$\mathcal{L}$ *iff it is valid in* dHL

- Containment imports **all** d$\mathcal{L}$ axioms to dH$\mathcal{L}$ once and for all, even when instantiated with proper dH$\mathcal{L}$ formulas.
- dH$\mathcal{L}$ axioms are single formulas, so each case of soundness only needs to show validity of one single formula.

# Concrete Reducibility

**Motivation:** What is the expressive power of d$\mathsf{H}\mathcal{L}$?

## Theorem (Concrete reducibility)

*Concrete* dHL *(i.e. without US symbols) reduces to concrete* d$\mathcal{L}$*.*
*There exists an effective reduction* $T : dH\mathcal{L} \to \mathrm{d}\mathcal{L}$ *such that when* $\phi \in dH\mathcal{L}$ *is concrete,* $T(\phi) \in \mathrm{d}\mathcal{L}$ *is valid iff* $\phi$ *is.*

## Proposition (Complexity of $T$)

$T$ *increases size quadratically, i.e.,* $|T(\phi)| \in \Theta(|\phi|^2)$ *for concrete* $\phi$*.*

**Implication:** $T$ cannot reduce axioms or certain advanced proof techniques. Reduction likely to bloat proofs in practice.

# Takeaways

- Info. flow analysis only as good as the model
- Hybrid models enable expressive CPS flows
- Logic d$\mathcal{HL}$ provides HDIF analysis.
- Hybrid logic (+ Uniform Substitution) provides clean foundation, High-level relational rules are derived
- Smart-grid example shows promise for practical applications
- **Future Work:** Hybrid logic as a broader foundation for hyperproperties, compare with other relational systems
- **Future Work:** Implementation to enable large-scale proofs