Verifying Hyperproperties of Hardware Systems

Bernd Finkbeiner
Saarland University

Markus N. Rabe
UC Berkeley

based on joint work with Michael R. Clarkson, Christopher Hahn, Masoud Koleini, Kristopher K. Micinski, and César Sánchez

FMCAD'16 Tutorial
Mountain View
October 3, 2016

thetechnews.com/2016/09/25/data-leak-yahoo-500-million Suchen

The Technews

Data leak at Yahoo: 500 million accounts stolen

SECURITY

SEPTEMBER 25, 2016

STELLA REYNELL

RELATED ITEMS DATA FEATURED YAHOO YAHOO ACCOUNTS STOLEN YAHOO DATA BREACH YAHOO DATA LEAK

2 SHARES

For months there were rumors about a data breach at Yahoo. Thursday, the Internet company announced that data from 500 million users were stolen. It thus seems to be the biggest data breach ever.

The cyber-attack would have been done at the end of 2014. The stolen data includes email addresses, phone numbers, dates of birth, encrypted passwords, and in some cases security

TAP

Major Incidents in Information Security

- Heartbleed **4.5m patient records leaked**

```
if (1 + 2 + payload + 16 > s->s3->rrec.length)
    return 0;
```

- Goto Fail **encryption of >300M devices broken**

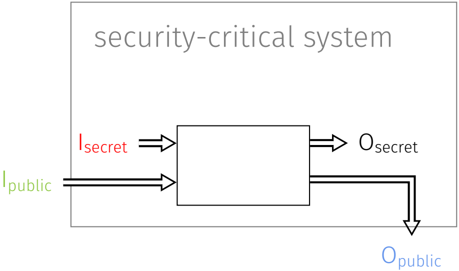
```
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
   goto fail;
   goto fail;
```

- Shellshock **web servers attackable for 22 years**

```
parse_and_execute (temp_string, name, SEVAL_NONINT|SEVAL_NOHIST);
```

Embedded Systems / Hardware Security

Information-flow control



Public output should only depend on public input.

Typical information-flow property: Noninterference

$$\forall t, t' \in \mathit{Traces}(K) : \; t =_{I_{public}} t' \;\Rightarrow\; t =_{O_{public}} t'$$

Hyperproperties

Clarkson&Schneider'10:
Hyperproperty H: a set of sets of traces

System K satisfies H iff $Traces(K) \in H$.

Hyperproperties

Clarkson&Schneider'10:
Hyperproperty H: a set of sets of traces

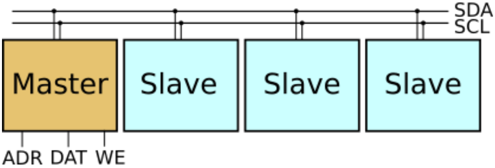
System K satisfies H iff $Traces(K) \in H$.

Many information-flow properties can be formalized as hyperproperties.

Noninterference as hyperproperty:

$$\{T \subseteq 2^{Traces} \mid \forall t, t' \in T : t =_{I_{public}} t' \;\Rightarrow\; t =_{O_{\text{public}}} t'\}$$

Case Study 1: Information flow in the I2C Bus



- Under which circumstances can information flow from the inputs through the Master to the bus (and vice versa)?
- Is there an expiration date for information?

Case study 2: Symmetry in Protocols

while ($true$) {
(1) choosing[i] = $true$;
(2) number[i] = max(number)+1;
(3) choosing[i] = $false$;
(4) **for** (int j=0; $j < n$; j++) {
    (5) **while** (choosing[j]) { ; }
    (6) **while** ($j \neq i \wedge$ number[j] $\neq 0 \wedge$ (number[j],j) $<$ (number[i],i)) { ; }
    }
(7) **critical**
(8) number[i] = 0;
(9) **non-critical**
}

- Are the clients treated symmetrically?

Case study 3: Error-resistant codes

Different encoders from OpenCores.org.

- 8bit-10bit encoder, decoder
- Huffman encoder
- Hamming encoder

- Do codes for distinct inputs have at least Hamming distance d?

Automatic analysis techniques

- Security type systems
- Program analysis
- Dynamic approaches/taint tracking

Common problem: single-property techniques

Automatic analysis techniques

- Security type systems
- Program analysis
- Dynamic approaches/taint tracking

Common problem: single-property techniques

This tutorial:
A unifying framework for the analysis of hyperproperties

Overview

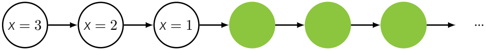
Part I

HyperLTL

Temporal logics for information security?

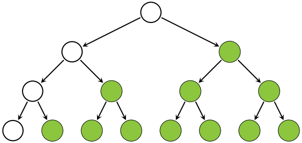
LTL: Specifies computations

Example: $\mathsf{F}\mathsf{G}\; x = 0$ “from some point on x is 0”



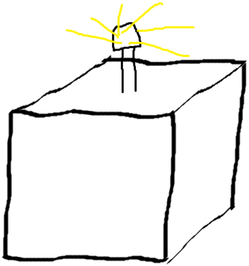
CTL/CTL*: Specifies computation trees

Example: $\mathsf{A}\mathsf{G}\mathsf{E}\mathsf{F}\; x = 0$ “x may always become 0 in the future”



A Simple Information-flow Policy

“All executions have the light on at the same time.”



“For all pairs of executions and all points in time, the light is on on the one execution iff it is on on the other execution.”

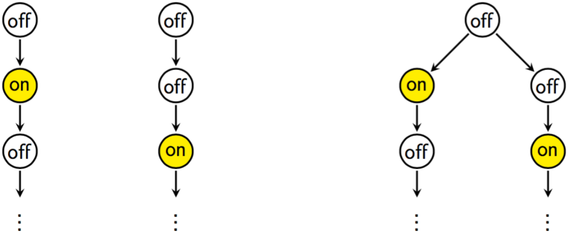
Information flow properties compare multiple executions!

LTL?

Syntax: $\varphi ::= a_\pi \mid X\psi \mid G\psi \mid F\psi \mid \psi U \psi \mid \psi W \psi$

Semantics: $K \models \varphi$ iff $\mathit{Traces}(K) \subseteq \mathit{Traces}(\varphi)$

"All executions have the light on at the same time."

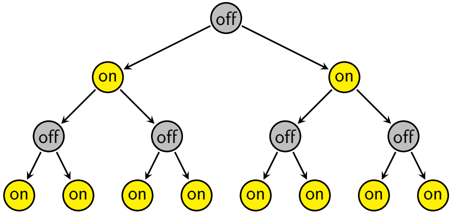


CTL*?

Syntax: $\varphi ::= a \mid A\varphi \mid E\varphi \mid X\varphi \mid G\varphi \mid \varphi U \varphi \mid \ldots$

Semantics: $K \models A\varphi$ iff for all $p \in \mathrm{Paths}(K) : p \models \varphi$

"All executions have the light on at the same time." $AA\varphi$ **?**

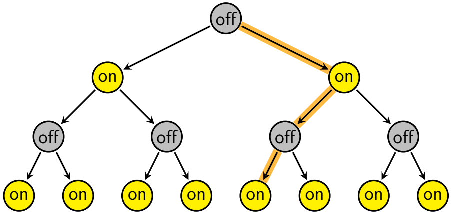


CTL*?

Syntax: $\varphi ::= a \mid A\varphi \mid E\varphi \mid X\varphi \mid G\varphi \mid \varphi U \varphi \mid \ldots$

Semantics: $K \models A\varphi$ iff for all $p \in \mathrm{Paths}(K) : p \models \varphi$

"All executions have the light on at the same time." $AA\varphi$ **?**

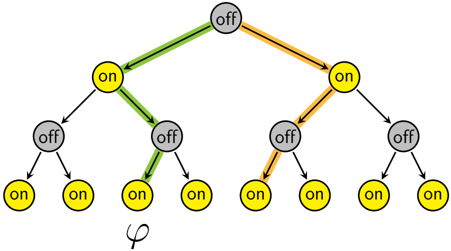


CTL*?

Syntax: $\varphi ::= a \mid \mathsf{A}\varphi \mid \mathsf{E}\varphi \mid \mathsf{X}\varphi \mid \mathsf{G}\varphi \mid \varphi \mathsf{U} \varphi \mid \ldots$

Semantics: $K \models \mathsf{A}\varphi$ iff for all $p \in \mathrm{Paths}(K) : p \models \varphi$

"All executions have the light on at the same time." $\mathsf{A}\mathsf{A}\varphi$ **?**



HyperLTL

Quantifiers with trace variables: $\forall\pi.\varphi \quad \exists\pi.\varphi$

Syntax:

$$\varphi ::= \forall\pi.\varphi \mid \exists\pi.\varphi \mid \psi$$
$$\psi ::= a_\pi \mid \mathsf{X}\psi \mid \mathsf{G}\psi \mid \mathsf{F}\psi \mid \psi\mathsf{U}\psi \mid \psi\mathsf{W}\psi$$

HyperLTL: Start with a quantifier prefix, then quantifier-free

HyperLTL

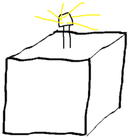
Quantifiers with trace variables: $\forall\pi.\varphi \quad \exists\pi.\varphi$

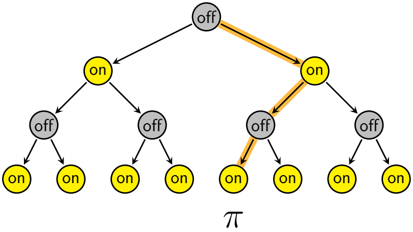
Syntax: $\varphi ::= \forall\pi.\varphi \mid \exists\pi.\varphi \mid \psi$
$\psi ::= a_\pi \mid \mathsf{X}\psi \mid \mathsf{G}\psi \mid \mathsf{F}\psi \mid \psi\mathsf{U}\psi \mid \psi\mathsf{W}\psi$

HyperLTL: Start with a quantifier prefix, then quantifier-free

"All executions have the light on at the same time."

$$\forall\pi.\forall\pi'.\ \mathsf{G}(\mathsf{on}_\pi \leftrightarrow \mathsf{on}_{\pi'})$$





HyperLTL

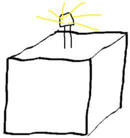
Quantifiers with trace variables: $\forall\pi.\varphi \quad \exists\pi.\varphi$

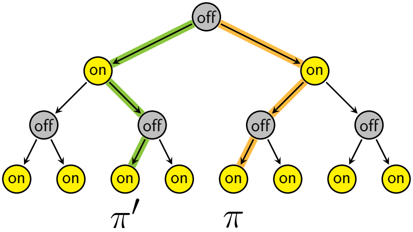
Syntax: $\varphi ::= \forall\pi.\varphi \mid \exists\pi.\varphi \mid \psi$
$\psi ::= a_\pi \mid \mathsf{X}\psi \mid \mathsf{G}\psi \mid \mathsf{F}\psi \mid \psi\mathsf{U}\psi \mid \psi\mathsf{W}\psi$

HyperLTL: Start with a quantifier prefix, then quantifier-free

"All executions have the light on at the same time."

$$\forall\pi.\forall\pi'.\ \mathsf{G}(\mathrm{on}_\pi \leftrightarrow \mathrm{on}_{\pi'})$$





HyperLTL

Quantifiers with trace variables: $\forall\pi.\varphi \quad \exists\pi.\varphi$

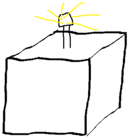
Syntax: $\varphi ::= \forall\pi.\varphi \mid \exists\pi.\varphi \mid \psi$

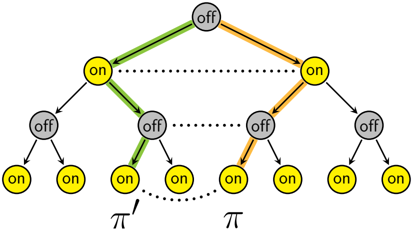
$\psi ::= a_\pi \mid \mathsf{X}\psi \mid \mathsf{G}\psi \mid \mathsf{F}\psi \mid \psi\mathsf{U}\psi \mid \psi W\psi$

HyperLTL: Start with a quantifier prefix, then quantifier-free

"All executions have the light on at the same time."

$$\forall\pi.\forall\pi'.\ \mathsf{G}(\mathrm{on}_\pi \leftrightarrow \mathrm{on}_{\pi'})$$





Semantics

$$
\begin{array}{lll}
\Pi \models_T a_\pi & \text{iff} & a \in \Pi(\pi)(0) \\
\Pi \models_T \mathsf{G}\varphi & \text{iff} & \forall i \geq 0 :\ \Pi[i,\infty] \models_T \varphi \\
\Pi \models_T \forall \pi.\ \varphi & \text{iff} & \forall t \in T :\ \Pi[\pi \mapsto t] \models_T \varphi
\end{array}
$$

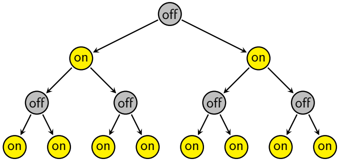
Semantics given with respect to a set of traces T and a trace environment $\Pi : Vars \to T$

A Kripke structure K satisfies a HyperLTL formula φ iff $\emptyset \models_{Traces(K)} \varphi$

Semantics

$$\begin{array}{lll} \Pi \models_T a_\pi & \text{iff} & a \in \Pi(\pi)(0) \\ \Pi \models_T \mathsf{G}\varphi & \text{iff} & \forall i \geq 0 : \ \Pi[i,\infty] \models_T \varphi \\ \Pi \models_T \forall \pi.\ \varphi & \text{iff} & \forall t \in T : \ \Pi[\pi \mapsto t] \models_T \varphi \end{array}$$

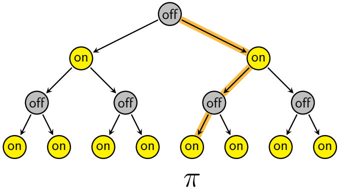
"All executions have the light on at the same time."



Semantics

$$\begin{array}{lll} \Pi \models_T a_\pi & \text{iff} & a \in \Pi(\pi)(0) \\ \Pi \models_T \mathrm{G}\varphi & \text{iff} & \forall i \geq 0 : \ \Pi[i,\infty] \models_T \varphi \\ \Pi \models_T \forall \pi.\, \varphi & \text{iff} & \forall t \in T : \ \Pi[\pi \mapsto t] \models_T \varphi \end{array}$$

"All executions have the light on at the same time."



1. $\{\pi \mapsto p\} \models_K \forall \pi'.\mathrm{G}(\mathrm{on}_\pi \leftrightarrow \mathrm{on}_{\pi'})$

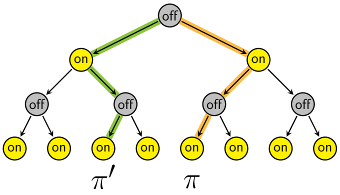
Semantics

$$\Pi \models_T a_\pi \quad \text{iff} \quad a \in \Pi(\pi)(0)$$

$$\Pi \models_T \mathsf{G}\varphi \quad \text{iff} \quad \forall i \geq 0 : \ \Pi[i,\infty] \models_T \varphi$$

$$\Pi \models_T \forall \pi.\ \varphi \quad \text{iff} \quad \forall t \in T : \ \Pi[\pi \mapsto t] \models_T \varphi$$

"All executions have the light on at the same time."



1. $\{\pi \mapsto p\} \models_K \forall \pi'.\mathsf{G}(\mathsf{on}_\pi \leftrightarrow \mathsf{on}_{\pi'})$
2. $\{\pi \mapsto p, \pi' \mapsto p'\} \models_K \mathsf{G}(\mathsf{on}_\pi \leftrightarrow \mathsf{on}_{\pi'})$

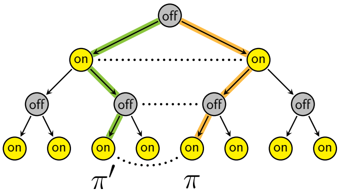
Semantics

$$\Pi \models_T a_\pi \quad \text{iff} \quad a \in \Pi(\pi)(0)$$

$$\Pi \models_T \mathsf{G}\varphi \quad \text{iff} \quad \forall i \geq 0 : \ \Pi[i,\infty] \models_T \varphi$$

$$\Pi \models_T \forall \pi.\ \varphi \quad \text{iff} \quad \forall t \in T : \ \Pi[\pi \mapsto t] \models_T \varphi$$

"All executions have the light on at the same time."



1. $\{\pi \mapsto p\} \models_K \forall \pi'.\mathsf{G}(\mathsf{on}_\pi \leftrightarrow \mathsf{on}_{\pi'})$
2. $\{\pi \mapsto p, \pi' \mapsto p'\} \models_K \mathsf{G}(\mathsf{on}_\pi \leftrightarrow \mathsf{on}_{\pi'})$
3. $\forall i \in \mathbb{N} : \ \{\pi \mapsto p[i,\infty], \pi' \mapsto p'[i,\infty]\} \models_K \mathsf{on}_\pi \leftrightarrow \mathsf{on}_{\pi'}$

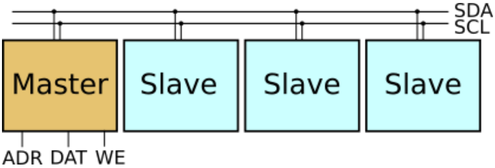
Full Semantics

$$
\begin{array}{lll}
\Pi \models_T a_\pi & \text{iff} & a \in \Pi(\pi)(0) \\
\Pi \models_T \mathsf{G}\varphi & \text{iff} & \forall i \geq 0 : \ \Pi[i,\infty] \models_T \varphi \\
\Pi \models_T \forall\pi.\ \varphi & \text{iff} & \forall t \in T : \ \Pi[\pi \mapsto t] \models_T \varphi \\
\Pi \models_T \neg\varphi & \text{iff} & \Pi \not\models_T \varphi \\
\Pi \models_T \varphi_1 \vee \varphi_2 & \text{iff} & \Pi \models_T \varphi_1 \text{ or } \Pi \models_T \varphi_2 \\
\Pi \models_T \mathsf{X}\varphi & \text{iff} & \Pi[1,\infty] \models_T \varphi \\
\Pi \models_T \mathsf{F}\varphi & \text{iff} & \exists i \geq 0 : \ \Pi[i,\infty] \models_T \varphi \\
\Pi \models_T \varphi_1 \mathsf{U} \varphi_2 & \text{iff} & \text{there exists } i \geq 0 : \Pi[i,\infty] \models_T \varphi_2 \\
 & & \quad \text{and for all } 0 \leq j < i \text{ we have } \Pi[j,\infty] \models_T \varphi_1 \\
\Pi \models_T \varphi_1 W \varphi_2 & \text{iff} & \Pi \models_T \varphi_1 \mathsf{U} \varphi_2 \text{ or } \Pi \models_T \mathsf{G}\varphi_1
\end{array}
$$

Part II

Examples

Case Study 1: Information flow in the I2C Bus



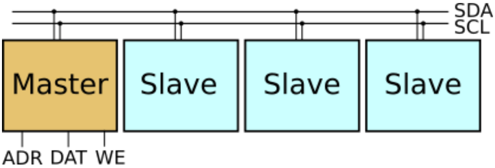
- Under which circumstances can information flow from the inputs through the Master to the bus?

$$\forall\pi.\forall\pi'.\ \mathsf{G}\,(\overline{\mathsf{DAT}}_{\pi} = \overline{\mathsf{DAT}}_{\pi'}) \Rightarrow\ \mathsf{G}\,(\mathsf{SDA}_{\pi} = \mathsf{SDA}_{\pi'})$$

$P_\pi = P_{\pi'}$ is defined as $\bigwedge_{a \in P} a_\pi \leftrightarrow a_{\pi'}$.

$\overline{P}_\pi = \overline{P}_{\pi'}$ is defined as $(I \setminus P)_\pi = (I \setminus P)_{\pi'}$.

Case Study 1: Information flow in the I2C Bus



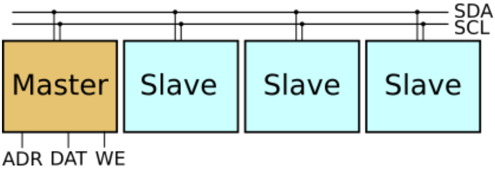
- Under which circumstances can information flow from the inputs through the Master to the bus?

$$\forall \pi . \forall \pi' .\ \mathrm{G}\,(\neg \mathrm{WE}_\pi \wedge \overline{\mathrm{DAT}}_\pi = \overline{\mathrm{DAT}}_{\pi'}) \Rightarrow\ \mathrm{G}\,(\mathrm{SDA}_\pi = \mathrm{SDA}_{\pi'})$$

$P_\pi = P_{\pi'}$ is defined as $\bigwedge_{a \in P} a_\pi \leftrightarrow a_{\pi'}$.

$\overline{P}_\pi = \overline{P}_{\pi'}$ is defined as $(I \setminus P)_\pi = (I \setminus P)_{\pi'}$.

Case Study 1: Information flow in the I2C Bus



- Is there an expiration date for information?

$$\forall\pi.\forall\pi'.\ (\overline{\mathrm{DAT}}_\pi = \overline{\mathrm{DAT}}_{\pi'}\ \mathrm{U}\ \mathrm{G}\,(I_\pi = I_{\pi'})) \Rightarrow \mathrm{F}\ \mathrm{G}\,(\mathrm{SDA}_\pi = \mathrm{SDA}_{\pi'}))$$

$P_\pi = P_{\pi'}$ is defined as $\bigwedge_{a \in P} a_\pi \leftrightarrow a_{\pi'}$.

$\overline{P}_\pi = \overline{P}_{\pi'}$ is defined as $(I \setminus P)_\pi = (I \setminus P)_{\pi'}$.

Variants of noninterference in HyperLTL

- Observational determinism [Zdancewich&Myers'03]

$$\forall\pi.\forall\pi'.\ lowIn_{\pi} = lowIn_{\pi'} \Rightarrow \mathsf{G}(lowOut_{\pi} = lowOut_{\pi'})$$

- Generalized noninterference [McCullough'88]

$$\forall\pi.\forall\pi'.\exists\pi''.\ \mathsf{G}(highIn_{\pi} = highIn_{\pi''}) \wedge \mathsf{G}(lowIn_{\pi'} = lowIn_{\pi''} \ \wedge \ lowOut_{\pi'} = lowOut_{\pi''})$$

- Noninference [McLean'94]

$$\forall\pi.\exists\pi'.\ \mathsf{G}(highIn_{\pi'}) \wedge \mathsf{G}(lowIn_{\pi} = lowIn_{\pi'} \ \wedge \ lowOut_{\pi} = lowOut_{\pi'})$$

Case study 2: Symmetry in Protocols

while ($true$) {
(1) choosing[i] = $true$;
(2) number[i] = max(number)+1;
(3) choosing[i] = $false$;
(4) **for** (int j=0; $j < n$; j++) {
  (5) **while** (choosing[j]) { ; }
  (6) **while** ($j \neq i \wedge$ number[j] $\neq 0 \wedge$ (number[j],j) $<$ (number[i],i)) { ; }
  }
(7) **critical**
(8) number[i] = 0;
(9) **non-critical**
}

- Are the clients treated symmetrically?

$$\forall\pi.\forall\pi'.\, \mathsf{G}\,(select\_0_\pi \leftrightarrow select\_1_{\pi'} \;\wedge\; select\_1_\pi \leftrightarrow select\_0_{\pi'}) \Rightarrow \mathsf{G}\,(critical\_0_\pi \leftrightarrow critical\_1_{\pi'} \;\wedge\; critical\_1_\pi \leftrightarrow critical\_0_{\pi'})$$

Case study 2: Symmetry in Protocols

while ($true$) {
(1) choosing[i] = $true$;
(2) number[i] = max(number)+1;
(3) choosing[i] = $false$;
(4) **for** (int j=0; $j < n$; j++) {
  (5) **while** (choosing[j]) { ; }
  (6) **while** ($j \neq i \wedge \text{number}[j] \neq 0 \;\wedge\; \begin{array}{c} f \wedge (\text{number}[j], j) < (number[i], i) \\ \vee \\ \neg f \wedge (\text{number}[j], i) < (number[i], j) \end{array}$) { ; }
  }
(7) **critical**
(8) number[i] = 0;
(9) **non-critical**
}

$$\forall\pi.\forall\pi'.\, \mathsf{G}\,(select\_0_\pi \leftrightarrow select\_1_{\pi'} \;\wedge\; select\_1_\pi \leftrightarrow select\_0_{\pi'} \;\wedge\; f_\pi \leftrightarrow \neg f_{\pi'}) \;\Rightarrow\; \mathsf{G}\,(critical\_0_\pi \leftrightarrow critical\_1_{\pi'} \;\wedge\; critical\_1_\pi \leftrightarrow critical\_0_{\pi'})$$

Case study 3: Error-resistant codes

Different encoders from OpenCores.org.

- 8bit-10bit encoder, decoder
- Huffman encoder
- Hamming encoder

- Do codes for distinct inputs have at least Hamming distance d?

$$\forall\pi.\forall\pi'.\ \mathsf{F}(\mathsf{DAT}_\pi \neq \mathsf{DAT}_{\pi'}) \ \Rightarrow\ \neg\mathsf{Ham}(d,\pi,\pi')$$

where we define:

$$\begin{aligned}\mathsf{Ham}(0,\pi,\pi') &= \ \mathsf{false}\\ \mathsf{Ham}(d,\pi,\pi') &= \ o_\pi = o_{\pi'}\ \mathsf{W}\ \big(o_\pi \neq o_{\pi'}\ \wedge\ \mathsf{X}\ \mathsf{Ham}(d-1,\pi,\pi')\big)\end{aligned}$$

Part III

Model Checking

Model Checking Alternation-free HyperLTL

- Reduction to emptiness of Büchi word automata
- Alphabet consists of tuples of sets of propositions.
- Projection handles quantifiers.

Model Checking Alternation-free HyperLTL

- Reduction to emptiness of Büchi word automata
- Alphabet consists of tuples of sets of propositions.
- Projection handles quantifiers.

Example: $\forall\pi.\forall\pi'.\ \mathrm{G}(\mathrm{i}_\pi \leftrightarrow \mathrm{i}_{\pi'}) \ \rightarrow\ \mathrm{G}(\mathrm{o}_\pi \leftrightarrow \mathrm{o}_{\pi'})$

Model Checking Alternation-free HyperLTL

- Reduction to emptiness of Büchi word automata
- Alphabet consists of tuples of sets of propositions.
- Projection handles quantifiers.

Example: $\forall\pi.\forall\pi'.\ \mathrm{G}(i_\pi \leftrightarrow i_{\pi'}) \ \rightarrow\ \mathrm{G}(o_\pi \leftrightarrow o_{\pi'})$

Negated: $\exists\pi.\exists\pi'.\ \mathrm{G}(i_\pi \leftrightarrow i_{\pi'}) \ \wedge\ \mathrm{F}(o_\pi \not\leftrightarrow o_{\pi'})$

Model Checking Alternation-free HyperLTL

- Reduction to emptiness of Büchi word automata
- Alphabet consists of tuples of sets of propositions.
- Projection handles quantifiers.

Example: $\forall\pi.\forall\pi'.\ \mathsf{G}(i_\pi \leftrightarrow i_{\pi'}) \ \rightarrow\ \mathsf{G}(o_\pi \leftrightarrow o_{\pi'})$

Negated: $\exists\pi.\exists\pi'.\ \underbrace{\mathsf{G}(i_\pi \leftrightarrow i_{\pi'}) \ \wedge\ \mathsf{F}(o_\pi \not\leftrightarrow o_{\pi'})}_{\mathcal{A} \text{ with alphabet } 2^{\mathsf{AP}} \times 2^{\mathsf{AP}}}$

Model Checking Alternation-free HyperLTL

- Reduction to emptiness of Büchi word automata
- Alphabet consists of tuples of sets of propositions.
- Projection handles quantifiers.

Example: $\forall\pi.\forall\pi'.\ \mathrm{G}(\mathrm{i}_\pi \leftrightarrow \mathrm{i}_{\pi'}) \ \rightarrow\ \mathrm{G}(\mathrm{o}_\pi \leftrightarrow \mathrm{o}_{\pi'})$

Negated: $\underbrace{\exists\pi.\,\exists\pi'.\ \underbrace{\mathrm{G}(\mathrm{i}_\pi \leftrightarrow \mathrm{i}_{\pi'}) \ \wedge\ \mathrm{F}(\mathrm{o}_\pi \not\leftrightarrow \mathrm{o}_{\pi'})}_{\mathcal{A} \text{ with alphabet } 2^{\mathrm{AP}} \times 2^{\mathrm{AP}}}}_{\mathcal{A}' \text{ with alphabet } 2^{\mathrm{AP}}}$

Model Checking Alternation-free HyperLTL

- Reduction to emptiness of Büchi word automata
- Alphabet consists of tuples of sets of propositions.
- Projection handles quantifiers.

Example: $\forall\pi.\forall\pi'.\ \mathsf{G}(\mathsf{i}_\pi \leftrightarrow \mathsf{i}_{\pi'}) \ \rightarrow\ \mathsf{G}(\mathsf{o}_\pi \leftrightarrow \mathsf{o}_{\pi'})$

Negated: $\underbrace{\exists\pi.\ \underbrace{\exists\pi'.\ \underbrace{\mathsf{G}(\mathsf{i}_\pi \leftrightarrow \mathsf{i}_{\pi'}) \ \wedge\ \mathsf{F}(\mathsf{o}_\pi \not\leftrightarrow \mathsf{o}_{\pi'})}_{\mathcal{A} \text{ with alphabet } 2^{\mathsf{AP}} \times 2^{\mathsf{AP}}}}_{\mathcal{A}' \text{ with alphabet } 2^{\mathsf{AP}}}}_{\mathcal{A}''' \text{ with 1-letter alphabet}}$

Model Checking General HyperLTL

Complexity depends on the quantifier alternation depth.

0. $\forall\pi.\forall\pi'.\psi$ PSPACE in $|\psi|$, NLOGSPACE in |K|
   - Observational determinism

Model Checking General HyperLTL

Complexity depends on the quantifier alternation depth.

0. $\forall\pi.\forall\pi'.\psi$ PSPACE in $|\psi|$, NLOGSPACE in |K|
   - Observational determinism
1. $\forall\pi.\exists\pi'.\psi$ EXPSPACE in $|\psi|$, PSPACE in |K|
   - Noninference
   - Generalized noninterference
2. ...

Rarely need more than one quantifier alternation!

Symbolic Model Checking for Circuits

- Alternation-free HyperLTL
- Clean extension of the circuit construction for LTL
- Leverages existing symbolic model checkers (e.g. ABC)

Reduction to Safety Property on Circuits

$$\forall\pi.\forall\pi'.\ \mathsf{G}\,(\mathsf{i}_\pi \leftrightarrow \mathsf{i}_{\pi'}) \;\Rightarrow\; \mathsf{G}\,(\mathsf{o}_\pi \leftrightarrow \mathsf{o}_{\pi'})$$

Reduction to Safety Property on Circuits

$$\forall\pi.\forall\pi'.\ \mathrm{G}\,(\mathrm{i}_\pi \leftrightarrow \mathrm{i}_{\pi'}) \ \Rightarrow\ \mathrm{G}\,(\mathrm{o}_\pi \leftrightarrow \mathrm{o}_{\pi'})$$

Negated: $\exists\pi.\exists\pi'.\ \mathrm{G}\,(\mathrm{i}_\pi \leftrightarrow \mathrm{i}_{\pi'}) \ \wedge\ \mathrm{F}\,(\mathrm{o}_\pi \not\leftrightarrow \mathrm{o}_{\pi'})$

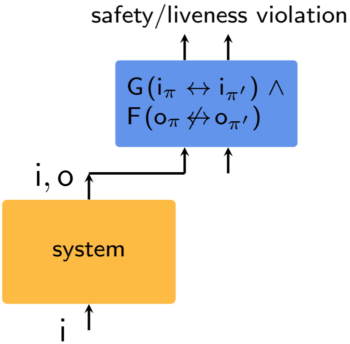
safety/liveness violation

$$\boxed{\begin{array}{l}\mathrm{G}\,(\mathrm{i}_\pi \leftrightarrow \mathrm{i}_{\pi'}) \wedge \\ \mathrm{F}\,(\mathrm{o}_\pi \not\leftrightarrow \mathrm{o}_{\pi'})\end{array}}$$

Reduction to Safety Property on Circuits

$$\forall\pi.\forall\pi'.\ \mathrm{G}(\mathrm{i}_\pi \leftrightarrow \mathrm{i}_{\pi'}) \Rightarrow \mathrm{G}(\mathrm{o}_\pi \leftrightarrow \mathrm{o}_{\pi'})$$

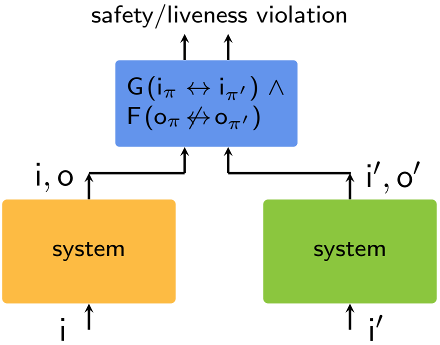
Negated: $\exists\pi.\exists\pi'.\ \mathrm{G}(\mathrm{i}_\pi \leftrightarrow \mathrm{i}_{\pi'}) \wedge \mathrm{F}(\mathrm{o}_\pi \not\leftrightarrow \mathrm{o}_{\pi'})$



Reduction to Safety Property on Circuits

$$\forall \pi. \forall \pi'.\ \mathsf{G}(i_\pi \leftrightarrow i_{\pi'}) \ \Rightarrow\ \mathsf{G}(o_\pi \leftrightarrow o_{\pi'})$$

Negated: $\exists \pi. \exists \pi'.\ \mathsf{G}(i_\pi \leftrightarrow i_{\pi'}) \ \wedge\ \mathsf{F}(o_\pi \not\leftrightarrow o_{\pi'})$



Implementation - MCHyper

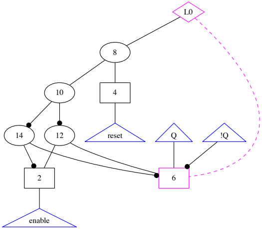
A transformation on Aiger circuits

Workflow

1. Convert VHDL/Verilog to Aiger
2. Run MCHyper with a formula and the circuit
3. Call a hardware model checker on the resulting circuit

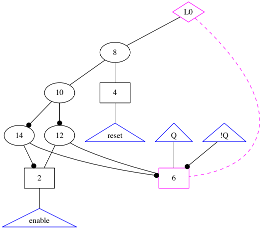
Tool website:
`https://www.react.uni-saarland.de/tools/mchyper/`

An Example Circuit

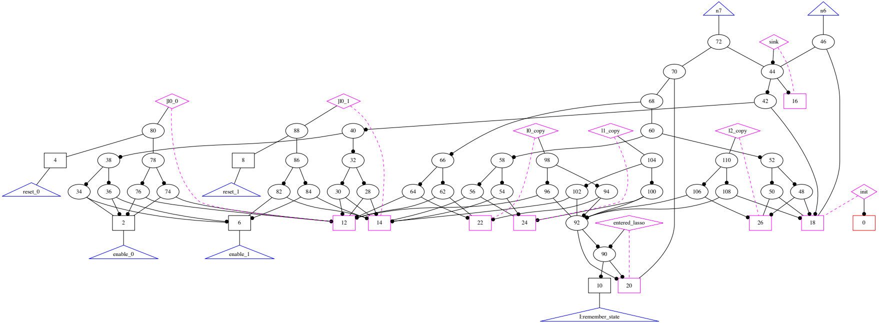


$$\forall \pi . \forall \pi' .\ \mathrm{G}(\mathrm{reset}_{\pi} \leftrightarrow \mathrm{reset}_{\pi'}) \ \Rightarrow\ \mathrm{G}(\mathrm{Q}_{\pi} \leftrightarrow \mathrm{Q}_{\pi'})$$

An Example Circuit



$$\forall\pi.\forall\pi'.\ \mathrm{G}(\mathrm{reset}_\pi \leftrightarrow \mathrm{reset}_{\pi'}) \Rightarrow \mathrm{G}(\mathrm{Q}_\pi \leftrightarrow \mathrm{Q}_{\pi'})$$



Information flow - Experimental Data

		Model	#Latches	#Gates	Verification time in s			Result
					IC3	INT	BMC	
IF1	(NI1)	I2C Master	254	1207	95.17	1.13	0.07	×
IF2	(NI2)				53.08	1.16	0.08	×
IF3	(NI3)				168.96	1.38	-	✓
IF4	(NI4)				438.41	1.01	0.09	×
IF5	(NI5)				717.74	8.31	0.77	×
IF6	(NI6)				186.20	1.10	0.07	×
IF7	(NI7)				TO	6.82	0.55	×
IF8	(NI8)				1557.14	2.92	0.16	×
IF9	(NI2′)	Ethernet	21093	70837	TO	155.77	6.27	×

Symmetry in Protocols - Experimental Data

		Model	#Latches	#Gates	Verification time in s IC3	INT	BMC	Result
Sym1	(S1)	Bakery	46	1829	6.34	0.88	0.08	×
Sym2	(S2)				168.59	464.52	7.00	×
Sym3		Bakery.a	47	1588	69.12	TO	71.92	×
Sym4	(S3)	Bakery.a.n	47	1618	26.31	4.75	0.39	×
Sym5		Bakery.a.n.s	47	1532	66.41	TO	-	✓
Sym6	(S4)				16.83	TO	-	✓
Sym7	(S5)	Bakery.a.n.s.5proc	90	3762	97.45	TO	-	✓
Sym8	(S6)				13.59	TO	-	✓
Sym9	(S7)	Bakery.a.n.s.7proc	136	6775	312.53*	TO	-	✓

Error Correcting Codes - Experimental Data

		Model	#Latches	#Gates	Verification time in s IC3	INT	BMC	Result
Huff1	(HD1)	Huffman_enc	19	571	3.08	37.19	-	✓
Huff2	(HD2)				0.62	0.09	0.02	×
8b10b_1	(HD1)	8b10b_enc	39	271	0.32	0.09	0.02	×
8b10b_2	(HD1′)				1.19	9.06	-	✓
8b10b_3	(HD2′)				0.03	0.04	0.02	×
8b10b_4	(HD1″)	8b10b_dec	19	157	0.05	0.09	-	✓
Hamm1	($HD1_1$)	Hamming_enc	27	47	0.02	0.04	0.02	×
Hamm2	($HD1_2$)				0.02	0.03	0.02	×
Hamm3	($HD1_3$)				0.03	0.04	0.02	×
Hamm3’	($HD1'_3$)				7.34	0.18	-	✓
Hamm4	($HD1_4$)				66.93	0.10	-	✓
Hamm5	($HD2_1$)				11.83	1.31	-	✓
Hamm6	($HD2_2$)				14.44	0.78	-	✓
Hamm7	(HD3)				12.23	1.25	-	✓

Part IV

Satisfiability

Satisfiability of HyperLTL

HyperLTL-SAT is the problem to decide whether there exists a non-empty trace set T satisfying a HyperLTL formula φ.

Application: Two versions of Observational Determinism:

- $\forall \pi. \forall \pi'. \mathsf{G}\,(I_\pi = I_{\pi'}) \rightarrow \mathsf{G}\,(O_\pi = O_{\pi'})$
- $\forall \pi. \forall \pi'. (O_\pi = O_{\pi'})\,\mathsf{W}\,(I_\pi \neq I_{\pi'})$

Which version is stronger?

Challenge

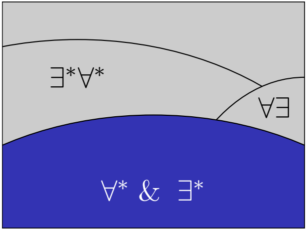
LTL Satisfiability Solving

- Translate LTL formula into Büchi automaton
- Check the automaton for emptiness
- PSPACE-complete

HyperLTL Satisfiability Solving

- A Hyperproperty is not necessarily ω-regular
- Standard automata approach cannot be applied

Solving HyperLTL-SAT



1. Alternation-free fragments ($\forall^*$ & $\exists^*$)
2. Alternation starting with existential quantifier ($\exists^*\forall^*$)
3. Alternation starting with universal quantifier ($\forall^*\exists^*$)

Existential Fragment

Theorem

$\exists^*$ HyperLTL-SAT is PSPACE-complete.

Example

$\exists\pi_0\exists\pi_1.\ \mathsf{G}a_{\pi_0} \land \mathsf{G}b_{\pi_0} \land \mathsf{G}c_{\pi_0} \land \mathsf{G}a_{\pi_1} \land \mathsf{G}\neg c_{\pi_1}$

Replace indexed atomic propositions with fresh atomic propositions.

$$\mathsf{G}a_0 \land \mathsf{G}b_0 \land \mathsf{G}c_0 \land \mathsf{G}a_1 \land \mathsf{G}\neg c_1$$

Existential Fragment

Theorem

$\exists^*$ HyperLTL-SAT is PSPACE-complete.

Example

$\exists \pi_0 \exists \pi_1 .\, \mathsf{G}\, a_{\pi_0} \wedge \mathsf{G}\, b_{\pi_0} \wedge \mathsf{G}\, c_{\pi_0} \wedge \mathsf{G}\, a_{\pi_1} \wedge \mathsf{G} \neg c_{\pi_1}$

Replace indexed atomic propositions with fresh atomic propositions.

$$\mathsf{G}\, a_0 \wedge \mathsf{G}\, b_0 \wedge \mathsf{G}\, c_0 \wedge \mathsf{G}\, a_1 \wedge \mathsf{G} \neg c_1$$
$$t : \{a_0, b_0, c_0, a_1\}^\omega$$

Existential Fragment

Theorem

$\exists^*$ HyperLTL-SAT is PSPACE-complete.

Example

$\exists \pi_0 \exists \pi_1 .\, \mathsf{G} a_{\pi_0} \wedge \mathsf{G} b_{\pi_0} \wedge \mathsf{G} c_{\pi_0} \wedge \mathsf{G} a_{\pi_1} \wedge \mathsf{G} \neg c_{\pi_1}$

Replace indexed atomic propositions with fresh atomic propositions.

$$\mathsf{G} a_0 \wedge \mathsf{G} b_0 \wedge \mathsf{G} c_0 \wedge \mathsf{G} a_1 \wedge \mathsf{G} \neg c_1$$
$$t : \{a_0, b_0, c_0, a_1\}^\omega$$

$$T = \{\{a, b, c\}^\omega, \{a\}^\omega\}$$

Universal Fragment

Theorem

$\forall^*$ HyperLTL-SAT is PSPACE-complete.

Example

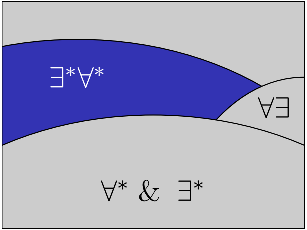
$$\forall\pi\forall\pi'.\mathsf{G}\, b_\pi \wedge \mathsf{G}\neg b_{\pi'} \qquad \equiv \qquad \mathsf{G}\, b \wedge \mathsf{G}\neg b$$

$\Downarrow\ \Downarrow$ $\qquad\qquad\qquad\qquad\qquad \Downarrow$

$t\quad t$ $\qquad\qquad\qquad\qquad\qquad$ unsatisfiable

Discard indexes from indexed propositions

Solving HyperLTL-SAT



1. Alternation-free fragments ($\forall^*$ & $\exists^*$)
2. Alternation starting with existential quantifier ($\exists^*\forall^*$)
3. Alternation starting with universal quantifier ($\forall^*\exists^*$)

$\exists^*\forall^*$ HyperLTL-SAT

Lemma

For every $\exists\pi_1 \ldots \exists\pi_n \forall\pi'_1 \ldots \forall\pi'_m.\varphi$ HyperLTL formula, there exists an equisatisfiable $\exists^*$ HyperLTL formula.

Example

$\exists\pi_0\exists\pi_1\forall\pi'_0\forall\pi'_1.\ (\mathsf{G}a_{\pi'_0} \wedge \mathsf{G}b_{\pi'_1}) \wedge (\mathsf{G}c_{\pi_0} \wedge \mathsf{G}d_{\pi_1})$

Unroll universal quantifiers

$$
\begin{aligned}
\exists\pi_0\exists\pi_1.\ & (\mathsf{G}a_{\pi_0} \wedge \mathsf{G}b_{\pi_0}) \wedge (\mathsf{G}c_{\pi_0} \wedge \mathsf{G}d_{\pi_1}) \\
& \wedge(\mathsf{G}a_{\pi_1} \wedge \mathsf{G}b_{\pi_0}) \wedge (\mathsf{G}c_{\pi_0} \wedge \mathsf{G}d_{\pi_1}) \\
& \wedge(\mathsf{G}a_{\pi_0} \wedge \mathsf{G}b_{\pi_1}) \wedge (\mathsf{G}c_{\pi_0} \wedge \mathsf{G}d_{\pi_1}) \\
& \wedge(\mathsf{G}a_{\pi_1} \wedge \mathsf{G}b_{\pi_1}) \wedge (\mathsf{G}c_{\pi_0} \wedge \mathsf{G}d_{\pi_1})
\end{aligned}
$$

Complexity of $\exists^*\forall^*$ HyperLTL-SAT

Theorem

Let n be the number of existential quantifiers and m be the number of universal quantifiers. $\exists^*\forall^*$ HyperLTL-SAT is EXPSPACE-complete in m.

- Unrolling results in formula of size $O(n^m)$.
- Hardness follows from an encoding of an EXPSPACE-bounded Turing machine in this fragment.

Application: Implication Checking of Quantifier-alternation-free Hyperproperties

ψ implies φ?
Check the negation $\psi \wedge \neg\varphi$ for unsatisfiability.

- If one formula is in the $\forall^*$ fragment and the other in the $\exists^*$ fragment, the resulting formula is alternation-free.
- If both ψ and φ are in the same fragment, then the resulting formula is in the $\exists^*\forall^*$ fragment.

Theorem

Implication Checking of quantifier-alternation-free HyperLTL formulas is EXPSPACE-complete.

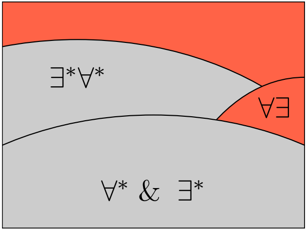
$\exists^*\forall^b$ HyperLTL-SAT

Theorem
Bounded $\exists^*\forall^b$ HyperLTL-SAT is PSPACE-complete.

Observation: In practice, many properties of interest quantify universally over pairs of traces

$$\forall\pi.\forall\pi'.\mathsf{G}(I_\pi = I_{\pi'}) \rightarrow \mathsf{G}(O_\pi = O_{\pi'})$$

Solving HyperLTL-SAT



1. Alternation-free fragments ($\forall^*$ & $\exists^*$)
2. Alternation starting with existential quantifier ($\exists^*\forall^*$)
3. Alternation starting with universal quantifier ($\forall^*\exists^*$)

The Power of ∀∃

$$\forall \pi \exists \pi'.\ a_{\pi'} \tag{1}$$
$$\wedge\, \mathsf{G}(a_\pi \rightarrow \mathsf{X}\mathsf{G}\neg a_\pi) \tag{2}$$
$$\wedge\, \mathsf{G}(a_\pi \rightarrow \mathsf{X} a_{\pi'}) \tag{3}$$

$$\begin{aligned} t_1 &:\ \{a\}(\{\})^\omega \\ t_2 &:\ \{\}\{a\}(\{\})^\omega \\ t_3 &:\ \{\}\{\}\{a\}(\{\})^\omega \\ &\dots \end{aligned}$$

→ Model has infinitely many traces.

Undecidability of $\forall\exists$ HyperLTL-SAT

Theorem

The satisfiability problem for any fragment of HyperLTL that contains the $\forall\exists$ formulas is undecidable.

- Undecidability follows from a reduction from Post's Correspondence Problem.

Summary HyperLTL-SAT

$\exists^*$	$\forall^*$	$\exists^*\forall^*$	Bounded $\exists^*\forall^*$	$\forall\exists$
PSpace-complete	PSpace-complete	EXPSpace-complete	PSpace-complete	undecidable

Part V

Beyond HyperLTL

Hyperproperties and branching-time logics

Observation:

Hyperproperties induce trace equivalence.

$$\forall K, K'.\ \ \mathrm{Traces}(K) = \mathrm{Traces}(K') \implies K \models H \leftrightarrow K' \models H$$

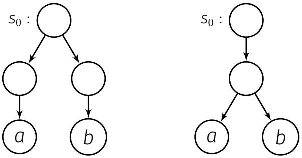
Hyperproperties and branching-time logics

Observation:

Hyperproperties induce trace equivalence.

$$\forall K, K'.\ \mathrm{Traces}(K) = \mathrm{Traces}(K') \implies K \models H \leftrightarrow K' \models H$$

Hyperproperties are not models for branching-time logics.



HyperCTL*

Syntax: $\varphi ::= a_\pi \mid \forall\pi.\varphi \mid \exists\pi.\varphi \mid X\varphi \mid G\varphi \mid \varphi U \varphi \mid \ldots$

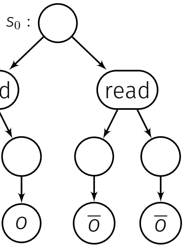
- HyperLTL: no quantifiers under temporal operators
- HyperCTL*: no restriction
- HyperCTL* with 1 path variable $\approx$ CTL*

What do we get beyond HyperLTL and CTL*?

```
bool y;
bool x = read(); // secret
output(y);
```

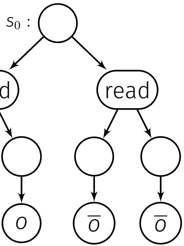
What do we get beyond HyperLTL and CTL*?

```
bool y;
bool x = read(); // secret
output(y);
```



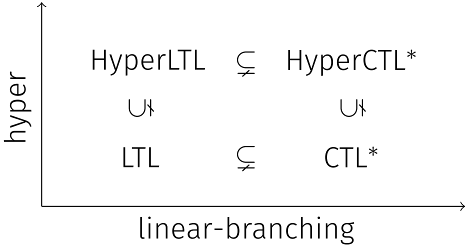
What do we get beyond HyperLTL and CTL*?

```
bool y;
bool x = read(); // secret
output(y);
```



$$\forall \pi.\ \mathsf{G}(\mathit{read}_\pi \rightarrow \forall \pi'.\ \mathsf{G}(o_\pi \leftrightarrow o_{\pi'}))$$

The Linear-Hyper-Branching Spectrum



- The induced process equivalence of HyperLTL is trace equivalence.

  Two systems with the same set of traces satisfy the same HyperLTL formulas.

- The induced process equivalence of HyperCTL* is bisimulation.

  Two bisimular systems satisfy the same HyperCTL formulas.*

HyperCTL* Model Checking

- Reduction to emptiness of Büchi word automata
- Alphabet consists of tuples of states.
- Projection handles quantifiers.
- Complementation handles quantifier alternations.

HyperCTL* Model Checking

- Reduction to emptiness of Büchi word automata
- Alphabet consists of tuples of states.
- Projection handles quantifiers.
- Complementation handles quantifier alternations.

Example: $\forall \pi.\, \mathrm{G}(a_\pi \Rightarrow \forall \pi'.\, \mathrm{G}(o_\pi \leftrightarrow o_{\pi'}))$

Negated: $\exists \pi.\, \mathrm{F}(a_\pi \wedge \exists \pi'.\quad \mathrm{F}(o_\pi \not\leftrightarrow o_{\pi'})\quad)$

HyperCTL* Model Checking

- Reduction to emptiness of Büchi word automata
- Alphabet consists of tuples of states.
- Projection handles quantifiers.
- Complementation handles quantifier alternations.

Example: $\forall\pi.\, \mathrm{G}(a_\pi \Rightarrow \forall\pi'.\, \mathrm{G}(o_\pi \leftrightarrow o_{\pi'}))$

Negated: $\exists\pi.\, \mathrm{F}(a_\pi \wedge \exists\pi'.\, \underbrace{\mathrm{F}(o_\pi \not\leftrightarrow o_{\pi'})}_{\mathcal{A} \text{ with alphabet } S\times S})$

HyperCTL* Model Checking

- Reduction to emptiness of Büchi word automata
- Alphabet consists of tuples of states.
- Projection handles quantifiers.
- Complementation handles quantifier alternations.

Example: $\forall \pi.\, \mathrm{G}\,(a_\pi \Rightarrow \forall \pi'.\, \mathrm{G}\,(o_\pi \leftrightarrow o_{\pi'}))$

Negated: $\exists \pi.\, \mathrm{F}\,(a_\pi \wedge \underbrace{\exists \pi'.\, \underbrace{\mathrm{F}\,(o_\pi \not\leftrightarrow o_{\pi'})}_{\mathcal{A} \text{ with alphabet } S \times S}}_{\mathcal{A}' \text{ with alphabet } S})$

HyperCTL* Model Checking

- Reduction to emptiness of Büchi word automata
- Alphabet consists of tuples of states.
- Projection handles quantifiers.
- Complementation handles quantifier alternations.

Example: $\forall\pi.\ \mathsf{G}(a_\pi \Rightarrow \forall\pi'.\ \mathsf{G}(o_\pi \leftrightarrow o_{\pi'}))$

Negated: $\exists\pi.\ \underbrace{\mathsf{F}(a_\pi \wedge \underbrace{\exists\pi'.\ \underbrace{\mathsf{F}(o_\pi \not\leftrightarrow o_{\pi'})}_{\mathcal{A}\text{ with alphabet } S\times S}}_{\mathcal{A}'\text{ with alphabet } S})}_{\mathcal{A}''\text{ with alphabet } S}$

HyperCTL* Model Checking

- Reduction to emptiness of Büchi word automata
- Alphabet consists of tuples of states.
- Projection handles quantifiers.
- Complementation handles quantifier alternations.

Example: $\forall\pi.\ \mathrm{G}(a_\pi \Rightarrow \forall\pi'.\ \mathrm{G}(o_\pi \leftrightarrow o_{\pi'}))$

Negated: $$\underbrace{\exists\pi.\ \underbrace{\mathrm{F}(a_\pi \wedge \underbrace{\exists\pi'.\ \underbrace{\mathrm{F}(o_\pi \not\leftrightarrow o_{\pi'})}_{\mathcal{A} \text{ with alphabet } S\times S}\ )}_{\mathcal{A}' \text{ with alphabet } S}}_{\mathcal{A}'' \text{ with alphabet } S}}_{\mathcal{A}''' \text{ with 1-letter alphabet}}$$

Part VI

Conclusions

Conclusions

- HyperLTL is a powerful tool
  for information security and beyond
  - Information-flow control
  - Symmetries in distributed systems
  - Error resistant codes
- Efficient model checking for alternation-free HyperLTL (non-elementary in general)
- Efficient satisfiability/implication/equivalence checking for alternation-free HyperLTL (undecidable in general)

Conclusions

- HyperLTL is a powerful tool
  for information security and beyond
  - Information-flow control
  - Symmetries in distributed systems
  - Error resistant codes
- Efficient model checking for alternation-free HyperLTL
  (non-elementary in general)
- Efficient satisfiability/implication/equivalence checking
  for alternation-free HyperLTL (undecidable in general)

Open problems

- HyperLTL on software
- Quantitative hyperproperties
- Specialized model checking algorithms

References

- Michael Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. **"Temporal Logics for Hyperproperties."** POST 2014
- Bernd Finkbeiner and Markus N. Rabe. **"The Linear-Hyper-Branching Spectrum of Temporal Logics."** it-Information Technology, 56, 2014.
- Bernd Finkbeiner, Markus N. Rabe, and César Sánchez. **"Algorithms for Model Checking HyperLTL and HyperCTL*."** CAV 2015
- Markus N. Rabe. **"A Temporal Logic Approach to Information-flow Control."** PhD Thesis. Saarland University, 2016.
- Bernd Finkbeiner and Christopher Hahn. **"Deciding Hyperproperties."** CONCUR 2016

Not covered in this tutorial:

- Bernd Finkbeiner, Helmut Seidl, and Christian Müller. **Specifying and Verifying Secrecy in Workflows with Arbitrarily Many Agents.** ATVA 2016.