Rich Counter-Examples for Temporal-Epistemic Logic Model Checking

Simon Busard Charles Pecheur
Université catholique de Louvain
Louvain-la-Neuve, Belgium

March 25, 2012

Running example: The dining cryptographers problem [1]

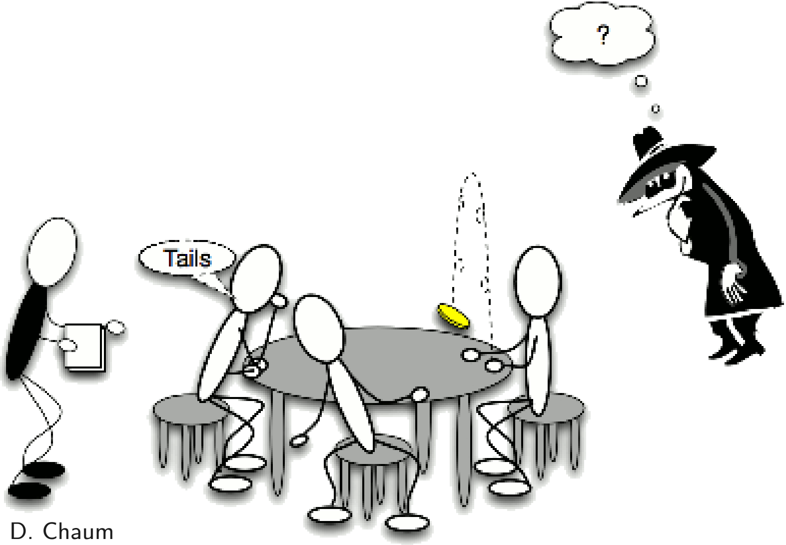
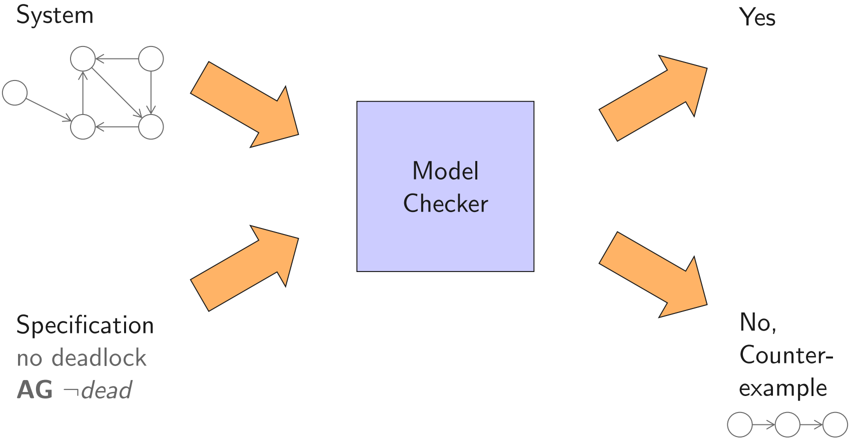


Image source: http://cgi.cse.unsw.edu.au/~mck/

[1] D. Chaum
The dining cryptographers problem: Unconditional sender and recipient untraceability. (1988)

Model Checking

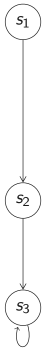


The problem

Most model checkers produce **linear counter-examples**.

"*If a did not pay, she will finally know that one of the others paid.*"

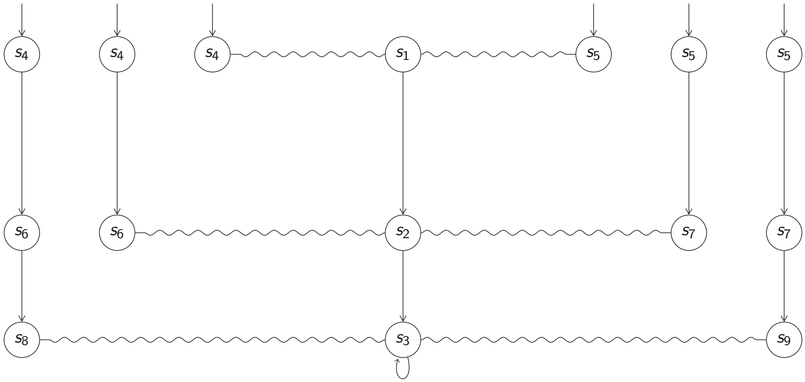
$\neg a.payer \implies \mathbf{AF}\ (\mathbf{K}_a\ b.payer \vee \mathbf{K}_a\ c.payer)$



A solution
Provide **full information**...

"*If a did not pay, she will finally know that one of the others paid.*"

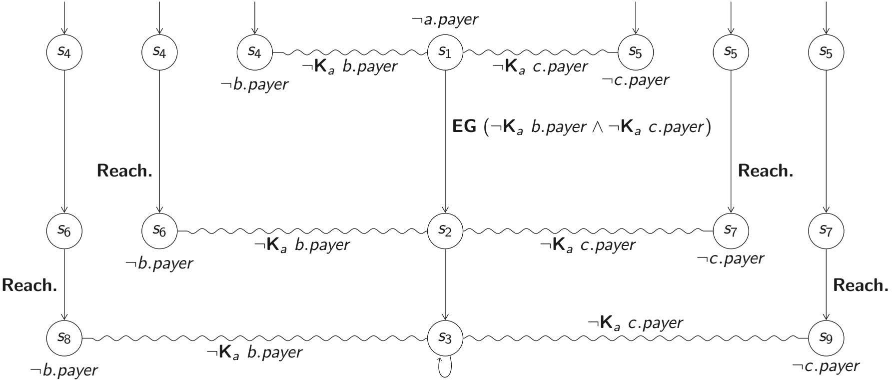
$\neg a.payer \implies \mathbf{AF}\ (\mathbf{K}_a\ b.payer \vee \mathbf{K}_a\ c.payer)$



A solution
... and **annotate** it.

" If a did not pay, she will finally know that one of the others paid."

$\neg a.payer \implies \mathbf{AF}\ (\mathbf{K}_a\ b.payer \vee \mathbf{K}_a\ c.payer)$



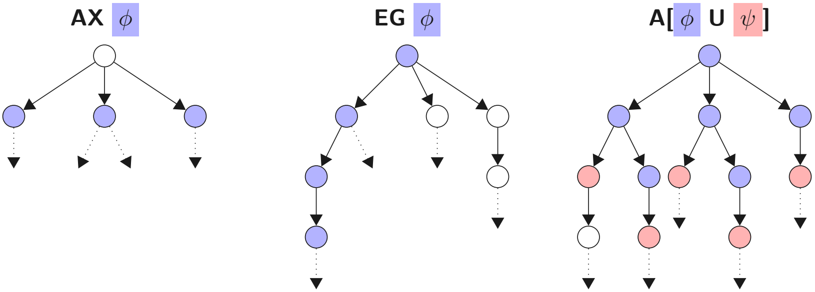
Outline

CTL, a branching temporal logic

Reasons about **system computation tree**.

Syntax Atomic propositions, logical connectives ($\wedge$, $\vee$, $\neg$), path quantifiers (**E**, **A**), temporal operators (**X**, **U**, **G**).

Semantics Interpreted over states of **Kripke structures**.



all successors satisfy ϕ

there exists a path where all states satisfy ϕ

for all paths, ϕ *is true up to a state satisfying* ψ

CTLK, CTL and knowledge

Reasons about **knowledge** and **time**.

Syntax CTL syntax + knowledge operators ($\mathbf{K}_{ag}$).

Semantics Interpreted over states of **Multi-Agent Systems**.
A state s satisfies $\mathbf{K}_{ag}\ \phi$ iff all states indistinguishable from s by ag satisfy ϕ.

"*If the count is even, everybody knows that NSA paid.*"
$even \implies (\mathbf{K}_a\ (\text{NSA paid}) \land \mathbf{K}_b\ (\text{NSA paid}) \land \mathbf{K}_c\ (\text{NSA paid}))$

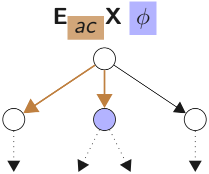
"*a never knows that b paid.*"
$\mathbf{AG}\ \neg\mathbf{K}_a\ b.payer$

ARCTL, Action-Restricted CTL

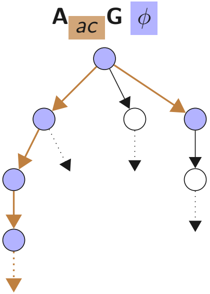
Reasons about **some paths** of system computation tree.

Syntax CTL syntax but **path quantifiers** hold an action formula ($\mathbf{E}_\alpha$, $\mathbf{A}_\alpha$).

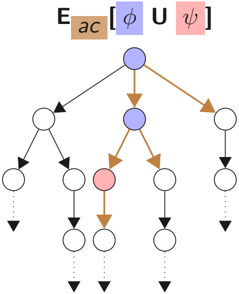
Semantics Interpreted over states of **Mixed Transition Systems**.



there exists a successor through action ac satisfying ϕ



all states of paths through ac actions satisfy ϕ



there exists a path through ac actions where ϕ is true up to a state satisfying ψ

From CTLK to ARCTL

CTLK and Multi-Agent Systems can be reduced to ARCTL and Mixed Transition Systems [2].

MAS to MTS: **temporal relation** is labelled with action *RUN*, **epistemic relations** are labelled with Agt_i.

CTLK to ARCTL: **temporal operators** are quantified over *RUN* actions, **epistemic operators** are quantified with Agt_i actions (+ **reachability**, i.e. a reverse temporal path).

We use an extension of NuSMV to model check CTLK.

[2] A. Lomuscio, C. Pecheur and F. Raimondi.
Automatic Verification of Knowledge and Time with NuSMV. (2007)

Outline

Tree-Like Annotated Counter-Examples

Branching, **annotated**, counter-examples.

States are annotated with sub-formulas they satisfy,
branches are annotated with sub-formulas they explain.

Note: a counter-example for ϕ is a witness for $\neg\phi$.

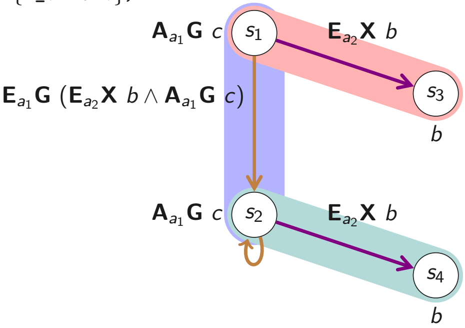
TLACE representation

$n := node($	a TLACE node is composed of
$s,$	a state,
$\{b \in AP\},$	a set of atomic propositions,
$\{\mathbf{E}_\alpha\ \pi : p\},$	a set of branches,
$\{\mathbf{A}_\alpha\ \pi\}$	a set of $\mathbf{A}_\alpha$ sub-formulas.
$)$	

A TLACE path p is simple: $\langle n, (a, n)^* \rangle$
or looping: $\langle n, (a, n)^*, a, loop(n) \rangle$

TLACE representation: example

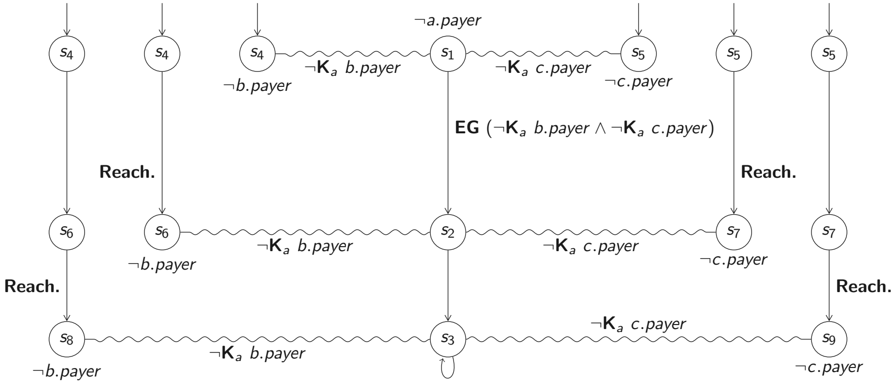
```
TLACE = node(s1, {}, {E_a1 G (E_a2 X b & A_a1 G c) : p}, {})

p = <n1, a1, n2, a1, loop(n2)>
n1 = node(s1, {}, {E_a2 X b : p1}, {A_a1 G c})
n2 = node(s2, {}, {E_a2 X b : p2}, {A_a1 G c})

p1 = <n11, a2, n12>
n11 = node(s1, {}, {}, {})
n12 = node(s3, {b}, {}, {})

p2 = <n21, a2, n22>
n21 = node(s2, {}, {}, {})
n22 = node(s4, {b}, {}, {})
```



Back to the dining cryptographers problem

"If a did not pay, she will finally know that one of the others paid."

$\neg a.payer \implies \mathbf{AF}\ (\mathbf{K}_a\ b.payer \vee \mathbf{K}_a\ c.payer)$



TLACE adequacy

A TLACE is **adequate** for $\mathcal{M}, s \models \phi$ iff it represents a witness explaining why $\mathcal{M}, s \models \phi$, i.e.

- TLACE paths are paths of $\mathcal{M}$;
- atomic propositions and action formulas are satisfied in $\mathcal{M}$;
- it has the structure of a witness for ϕ;
- annotations are coherent with ϕ.

TLACE fullness

Fullness: there exists an **adequate** TLACE witness for $\mathcal{M}, s \models \phi$ if and only if $\mathcal{M}, s \models \phi$.

Adequacy is not sufficient for fullness because TLACEs **do not explain $\mathbf{A}_\alpha$ operators**.

Adequacy is sufficient for fullness in the **existential fragment of ARCTL**.

TLACE generating algorithm

Works **recursively** over the **structure of the formula**.

For temporal operators,

1. get a **path** explaining the formula;
2. get a **TLACE** for each of its states;
3. **combine** these nodes into a new node.

TLACE generation: $\mathbf{E}_\alpha\mathbf{U}$ case ($\mathbf{E}_\alpha[\psi_1 \ \mathbf{U} \ \psi_2]$)

$$\langle s_0, a_1, ..., s_m \rangle \leftarrow \mathit{EaUexplain}(\mathcal{M}, s, \psi_1, \psi_2, \alpha)$$

$p \leftarrow \langle\rangle$
for $i \in 0..m-1$ **do**
    $p \leftarrow p + \langle \mathit{explain}(\mathcal{M}, s_i, \psi_1), a_{i+1} \rangle$
end
$p \leftarrow p + \langle \mathit{explain}(\mathcal{M}, s_m, \psi_2) \rangle\}$

return $\mathit{node}(s, \{\}, \{\mathbf{E}_\alpha[\psi_1 \ \mathbf{U} \ \psi_2] : p\}, \{\})$

Outline

Tools

- TLACEs can be large: $\mathcal{O}(|S|^{|\phi|})$ in worst case.
- Need ways to deal with that richness: **tool support**.

- NuSMV: extended to generate TLACEs.
- TLACEVisualizer: visualize and manipulate TLACEs.

NuSMV-ARCTL-TLACE

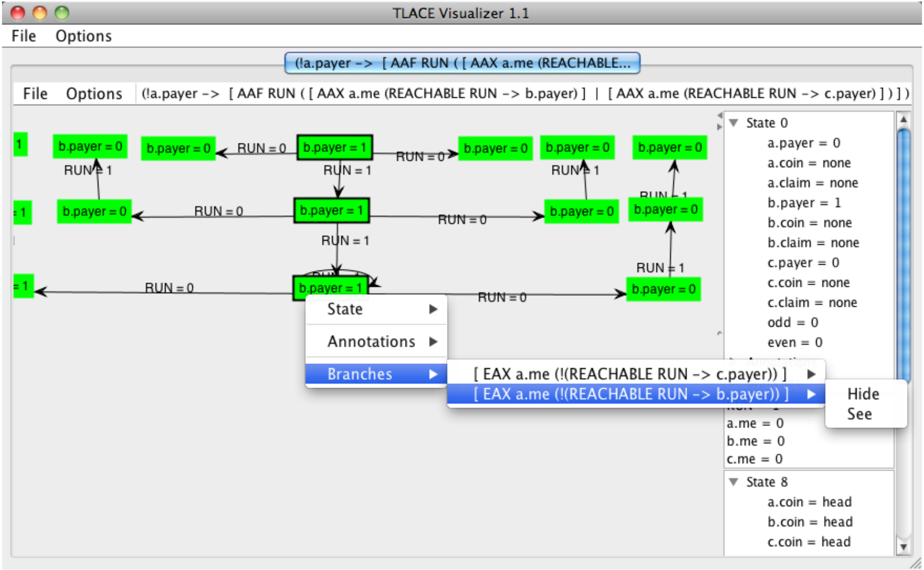
- NuSMV: **state-of-the-art** model checker.

- Modified to model check ARCTL [3].
  (CTLK transformed into ARCTL before model checking)

- Modified to **generate TLACEs**
  - TLACEs exported in XML.
  - Some options to **handle TLACE size**:
    explained temporal operators and maximum depth.

[3] C. Pecheur and F. Raimondi. *Symbolic model checking of logics with actions.* (2007)

TLACEVisualizer: graphical tool for visualizing and manipulating TLACEs



Outline

Rich Branching Logics

Tree-Like Annotated Counter-Examples

Tools

Conclusions and Perspectives

Richer counter-examples for rich logics

Contributions:

- **formalization** of tree-like annotated counter-examples;
- **characterization** of tree-like annotated counter-examples;
- design of an **algorithm** to generate them;
- **implementation** of this algorithm in NuSMV;
- design and implementation of **TLACEVisualizer** to visualize and manipulate TLACEs.

Perspectives: interactive witness generation

Deal with TLACEs size by generating counter-examples **interactively**.

1. User asks for partial explanation; system explains one sub-formula at a time.
2. Solution for $\mathbf{A}_\alpha$ operators explanation: the user plays a **game** against the system to understand why the property is satisfied.