FAIRNESS FOR INFINITE STATE SYSTEMS

Heidy Khlaaf[1] Byron Cook[1] Nir Piterman[2]

[1] University College London
[2] University of Leicester

FAIRNESS

- If a process requests a resource infinitely often, then it must be granted infinitely often (resource starvation).
- Verifying fairness:
  - Bridges the gap between trace-based and state-based reasoning, allowing us to prove things like fair-termination.
  - When proving state-based properties, fairness is used to model trace-based assumptions about the environment.

FAIRNESS

```
PPBlockInits();
while (i < Pdolen) {
    DName = PPMakeDeviceName(…);
    if (!DName) { break; }
    RtlInitUnicodeString(&deviceName, DName);
    status = IoCreateDevice(…);
    if (STATUS_SUCCESS != status) {
        Pdo[i] = NULL;
        if (STATUS_OBJECT_NAME_COLLISION == status) {
            ExFreePool(DName);
            num++;
            continue;
        }
        break;
    } else {
        i++;
    }
}
num = 0;
PPUnblockInits();
```

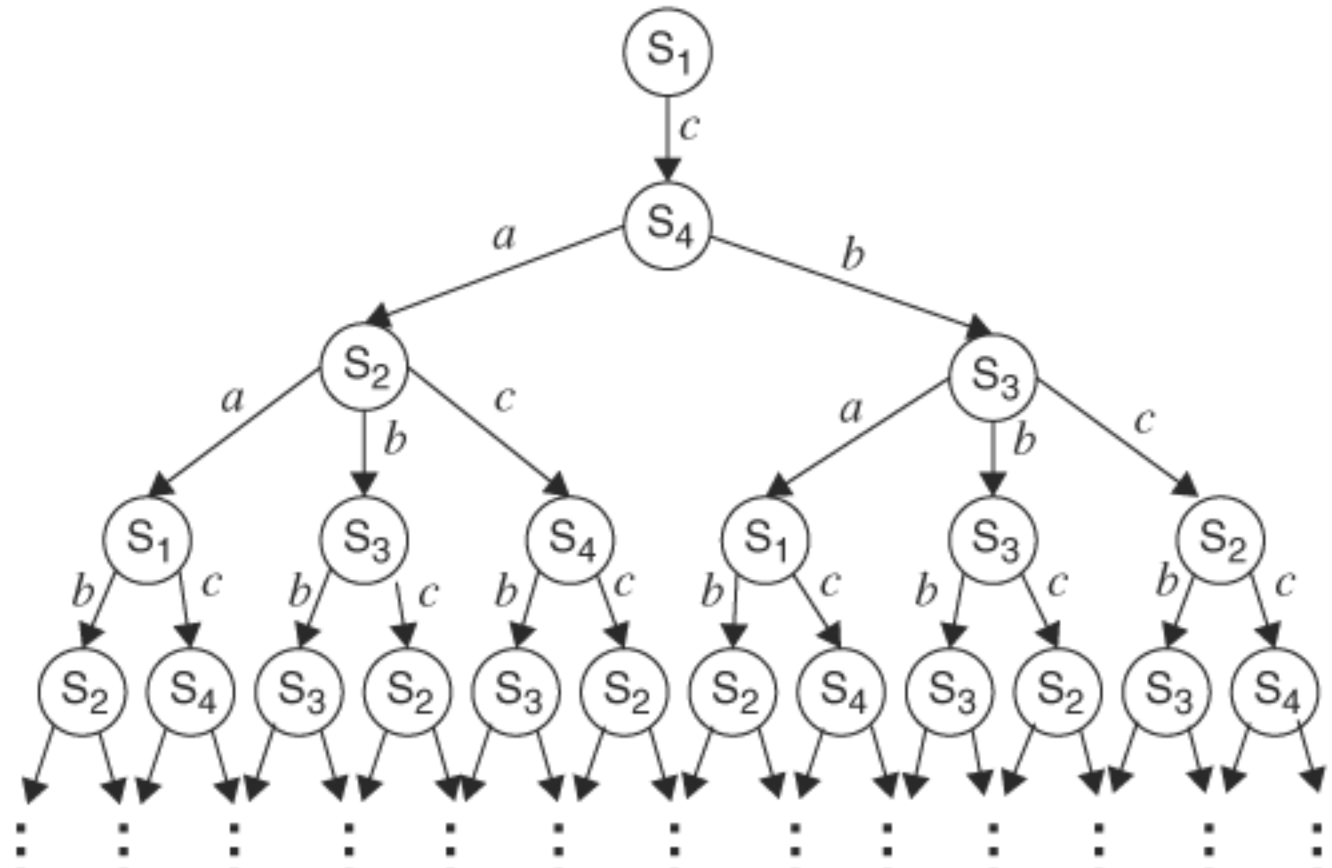
EMPLOYING FAIRNESS

- First known tool for symbolically proving fair-CTL properties of *infinite*-state programs.
- Solution is based on a reduction to existing techniques for fairness- free CTL model checking via prophecy variables.
    - Prophecy variables are auxiliary variables whose values are defined in terms of current program state and future behavior.

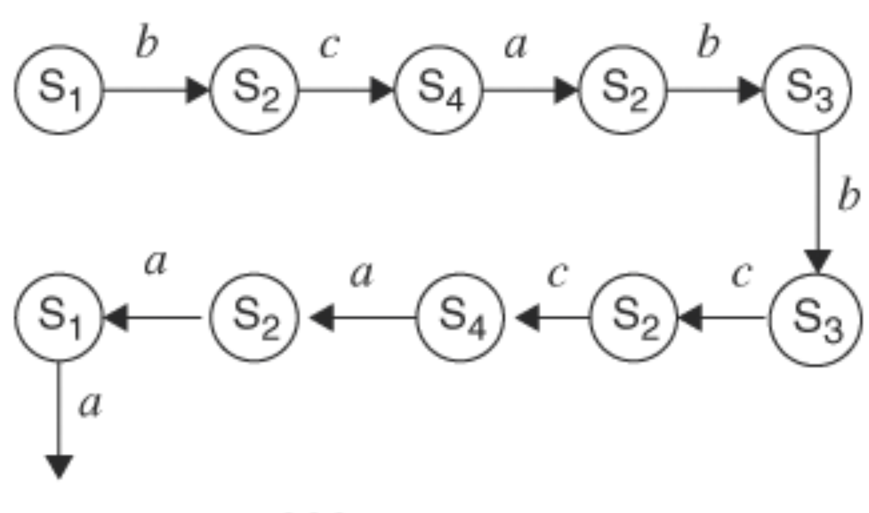
TEMPORAL LOGIC

- Logic reasoning about propositions qualified in terms of time.
- Used as a specification language as it encompasses safety, liveness, fairness, etc.
- Most commonly used sub-logics are CTL (state based) and LTL (trace based).

CTL VS LTL



CTL



LTL

CTL

- Reasoning about sets of states.
- Reasoning about non-deterministic (branching) programs.
- $\varphi ::= \alpha \mid \neg\alpha \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid AX\varphi \mid AF\varphi \mid A[\varphi W \varphi] \mid EX\varphi \mid EG\varphi \mid E[\varphi U \varphi]$
- A φ – All: φ has to hold on all paths starting from all initial states.
- E φ – Exists: there exists at least one path starting from all initial states where φ holds.

CTL

- X φ – Next: φ has to hold at the next state.
- G φ – Globally: φ has to hold on the all states along a path.
- F φ – Finally: φ eventually has to hold.
- φ_1 U φ_2 – Until: φ_1 has to hold at least until at some position φ_2 holds. φ_2 must be verified in the future.
- φ_1 W φ_2 – Weak until: φ_1 has to hold until φ_2 holds.

LTL

- Reasoning about sets of paths.
- Reasoning about concurrent programs.
- $\psi ::= \alpha \mid \psi \wedge \psi \mid \psi \vee \psi \mid G\psi \mid F\psi \mid [\psi W \psi] \mid [\psi U \psi]$.
- Properties expressed in the universal fragment of CTL (∀CTL) are easier to prove than LTL properties.

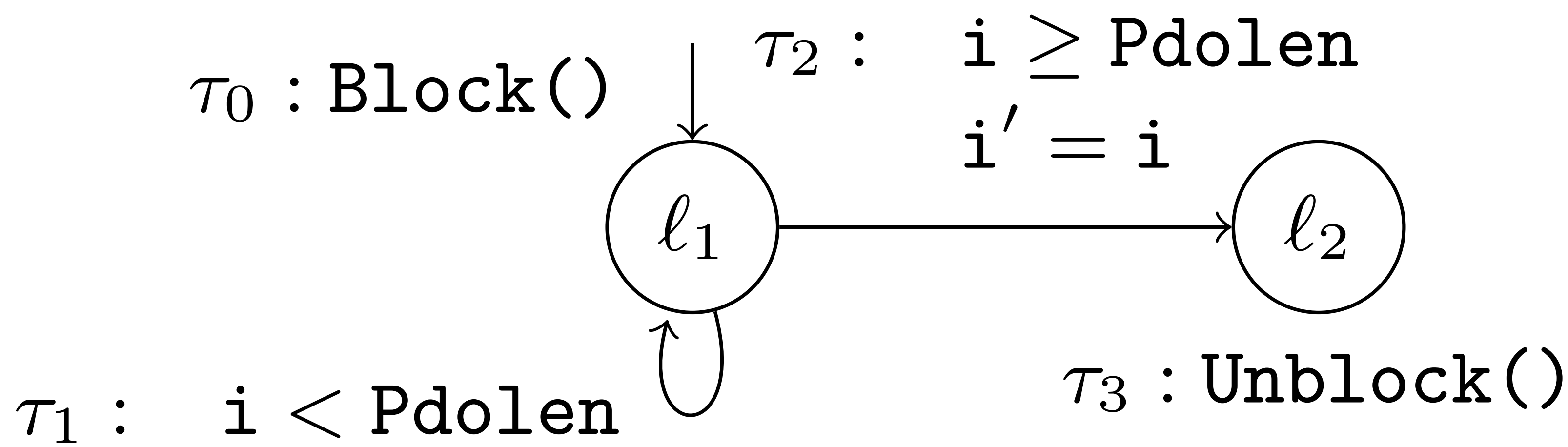
LTL

- Can naturally express fairness: GF $p \Rightarrow$ GF q.
    - Path based property not expressible in CTL.
    - When proving state-based CTL properties, we must often use fairness to model path-based assumptions about the environment.
    - When reasoning about concurrent environments, fairness is used to abstract away the scheduler.

FAIR LIVENESS AG (*BLOCK()* ⇒ AF *UNBLOCK()*)

```
PPBlockInits();
while (i < Pdolen) {
    DName = PPMakeDeviceName(…);
    if (!DName) { break; }
    RtlInitUnicodeString(&deviceName, DName);
    status = IoCreateDevice(…);
    if (STATUS_SUCCESS != status) {
        Pdo[i] = NULL;
        if (STATUS_OBJECT_NAME_COLLISION == status) {
            ExFreePool(DName);
            num++;
            continue;
        }
        break;
    } else {
        i++;
    }
}
num = 0;
PPUnblockInits();
```

FAIR LIVENESS



- Transition system contains a non-terminating execution.
- However, if we only allow fair executions, then it is fair-terminating given that there exists no infinite fair paths such that if τ_1 occurs infinitely often then so does τ_2.
- CTL can express liveness properties such as **AG(*Block()* ⇒ AF *unblock()*)** but not that it should hold only under fair paths.

FAIR CTL

- A transition system $M = (S, S_0, R, L)$ and a fairness condition $\Omega = (p,q)$ where $p,q \subseteq S$.
- An infinite path π is unfair under Ω if states from p occur infinitely often along π but states from q occur finitely often. Otherwise, π is fair.

FAIR CTL

- Fair CTL model checking restricts the checks to only fair paths:
    1. M, si $\models_{\Omega+}$ Aφ iff φ holds in ALL fair paths.
    2. M, si $\models_{\Omega+}$ Eφ iff φ holds in one or more fair paths.
- Idea: Reduce fair CTL to fairness-free CTL via prophecy variables.
- Use the prophecy to encode a partition of fair from unfair paths.

THE REDUCTION

$$\textsc{Fair}((S, S_0, R, L), (p, q)) \quad \triangleq \quad (S_\Omega, S^0_\Omega, R_\Omega, L_\Omega)$$

where

$$\begin{aligned} S_\Omega &= S \times \mathbb{N} \\ R_\Omega &= \{((s,n),(s',n')) \mid (s,s') \in R\} \wedge \left(\begin{array}{c} (\neg p \wedge n' \le n) \vee \\ (p \wedge n' < n) \vee \\ q \end{array} \right) \\ S^0_\Omega &= S^0 \times \mathbb{N} \\ L_\Omega(s,n) &= L(s) \end{aligned}$$

- n is decreased whenever a transition imposing $p \wedge n' < n$ is taken.
- Since $n \in N$, n cannot decrease infinitely often, enforcing the eventual invalidation of the transition $p \wedge n' < n$.
- R_Ω would only allow a transition to proceed if q holds or $\neg p \wedge n' \le n$ holds. That is, either q occurs infinitely often or p will occur finitely often.

TRANSFORMATION

$$R' = R \cup \{(s,s) \mid \ \forall s'.(s,s') \notin R\} \qquad L'(s) = \begin{cases} L(s) \cup \{t\}, & \text{if } \forall s'.(s,s') \notin R \\ L(s), & \text{otherwise} \end{cases}$$

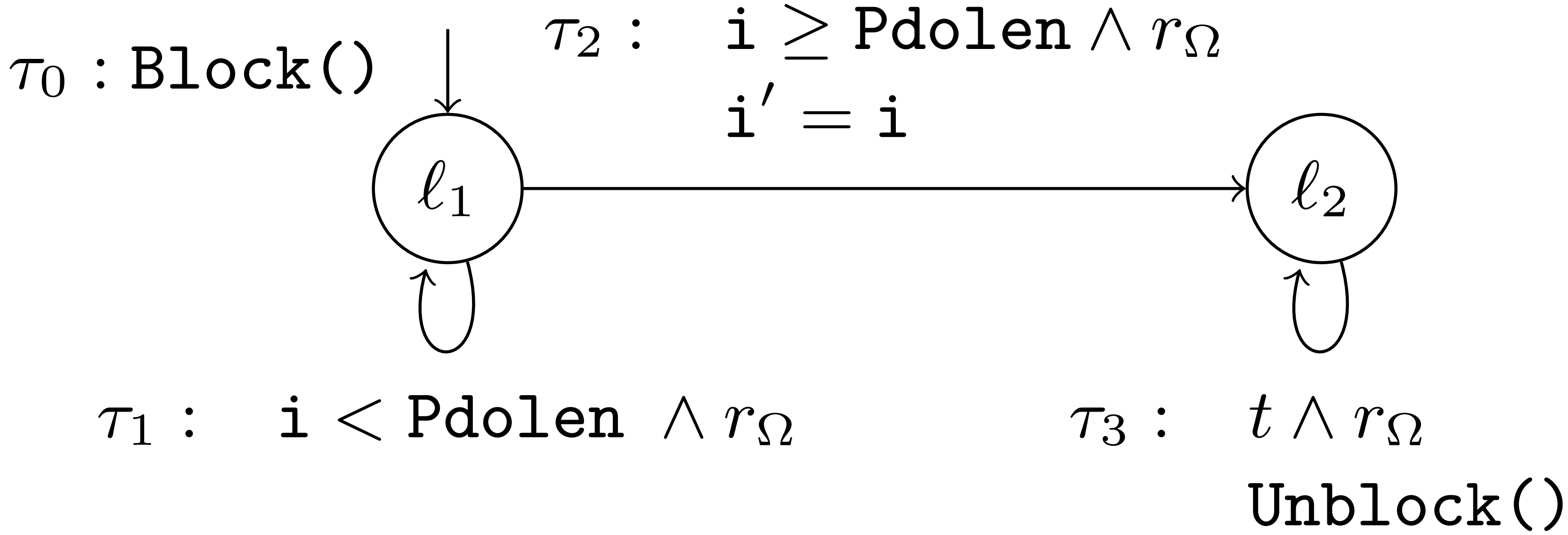
- *Fair(M,*Ω*)* can include finite paths that are prefixes of unfair infinite paths due to the wrong estimation of the number of *p*-s until *q*.
- Must ensure that these paths do not interfere with the validity of our model checking procedure.
- We distinguish between finite paths that occur in *M* and those introduced by our reduction.
- Add a self-loop with proposition *t* to mark all original "valid" termination states.

TRANSFORMATION

$$
\begin{aligned}
\textsc{Term}(\alpha, t) &::= \alpha \\
\textsc{Term}(\varphi_1 \wedge \varphi_2, t) &::= \textsc{Term}(\varphi_1, t) \wedge \textsc{Term}(\varphi_2, t) \\
\textsc{Term}(\varphi_1 \vee \varphi_2, t) &::= \textsc{Term}(\varphi_1, t) \vee \textsc{Term}(\varphi_2, t) \\
\textsc{Term}(\mathsf{AX}\varphi, t) &::= t \vee \mathsf{AX}(\textsc{Term}(\varphi, t)) \\
\textsc{Term}(\mathsf{AF}\varphi, t) &::= \mathsf{AF}\textsc{Term}(\varphi, t) \\
\textsc{Term}(\mathsf{A}[\varphi_1 \mathsf{W} \varphi_2], t) &::= \mathsf{A}[\textsc{Term}(\varphi_1, t)\ \mathsf{W}\ \textsc{Term}(\varphi_2, t)] \\
\textsc{Term}(\mathsf{EX}\varphi, t) &::= \neg t \wedge \mathsf{EX}(\textsc{Term}(\varphi, t)) \\
\textsc{Term}(\mathsf{EG}\varphi, t) &::= \mathsf{EG}\textsc{Term}(\varphi, t) \\
\textsc{Term}(\mathsf{E}[\varphi_1 \mathsf{U} \varphi_2], t) &::= \mathsf{E}[\textsc{Term}(\varphi_1, t)\ \mathsf{U}\ \textsc{Term}(\varphi_2, t)]
\end{aligned}
$$

- Adjust the CTL specification to accommodate for this change.
- $M \models_{\Omega+} \varphi \Leftrightarrow \mathit{Term}(M, t) \models_{\Omega+} \mathit{Term}(\varphi, t)$

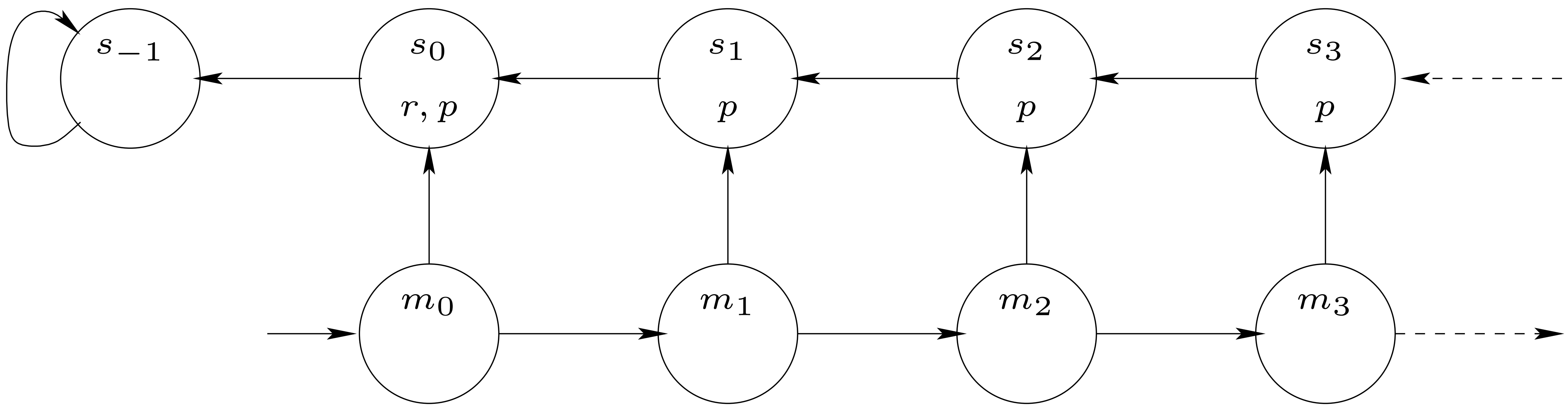
FAIR TERMINATION - REVISITED



$$r_\Omega : \{\ (\neg\tau_1 \wedge n' \leq n)\ \vee (\tau_1 \wedge n' < n)\ \vee \tau_2\ \} \wedge n \geq 0$$

- CTL property **AG(*lock()* ⇒ AF *unlock()*).**
- Strong fairness constraint **Ω** = (**τ**1, **τ**2).

ECTL



- $M, m_0 \models_{\Omega+} \mathrm{EG}(\neg p \wedge \mathrm{EF}\, r)$ for $\Omega = (p,q)$.
- From s_i there is a path that eventually reaches s_0, where it satisfies r, and then continues to s_{-1}, where p does not hold.
- The paths which satisfy $\mathrm{EG}(\neg p \wedge \mathrm{EF}\, r)$ are fair.
- However, system does not hold under $Fair(M,\Omega)$.

FAIR CTL

- As long as a new prophecy variable is introduced for each temporal sub-formula, the reduction can still be applied.
- Recurse over each sub-formula, and add a non-termination (E$\boldsymbol{\varphi}$) or termination (A$\boldsymbol{\varphi}$) clause, allowing us to ignore finite paths that are prefixes of unfair infinite-paths.
- Apply our reduction Fair(M, Ω) and run with φ on an existing CTL model checker which returns an assertion $\boldsymbol{a}$ characterizing the states in which φ holds.
- $E\varphi' \rightarrow \exists\, n \geq 0 \,.\, \boldsymbol{a}$
- $A\varphi' \rightarrow \forall\, n \geq 0 \,.\, \boldsymbol{a}$

FAIR CTL

```
let FAIRCTL(M, Ω, φ) : assertion =

    match(φ) with
    | Q φ1 OP φ2
    | φ1 bool_OP φ2 →
        a_φ1 = FAIRCTL(M, Ω, φ1);
        a_φ2 = FAIRCTL(M, Ω, φ2)
    | Q OP φ1 →
        a_φ1 = FAIRCTL(M, Ω, φ1)
    | α →
        a_φ1 = α

    match(φ) with
    | E φ1 U φ2 →
        φ' = E[a_φ1 U (a_φ2 ∧ ¬term)]
    | E Gφ1 →
        φ' = EG(a_φ1 ∧ ¬term)
    | E Xφ1 →
        φ' = EX(a_φ1 ∧ ¬term)
    | A φ1 W φ2 →
        φ' = A[a_φ1 W (a_φ2 ∨ term)]

    | A Fφ1 →
        φ' = AF(a_φ1 ∨ term)
    | A Xφ1 →
        φ' = AX(a_φ1 ∨ term)
    | φ1 bool_OP φ2 →
        φ' = a_φ1 bool_OP a_φ2
    | α →
        φ' = a_φ1

    M' = FAIR(M, Ω)
    a = CTL(M', φ')

    match(φ) with
    | E φ' →
        return ∃n ≥ 0 . a
    | A φ' →
        return ∀n ≥ 0 . a
    | _ →
        return a
```

- *FairCTL(M,Ω,φ)* employs an existing CTL model checker and the reduction Fair(M, Ω). An assertion characterizing the states in which **φ** holds under the fairness constraint **Ω** is returned.

EXPERIMENTS

Program	LOC	Property	FC	Time(s)	Result
WDD1	20	AG(BlockInits() ⇒ AF UnblockInits())	Yes	14.4	✓
WDD1	20	AG(BlockInits() ⇒ AF UnblockInits())	No	2.1	χ
WDD2	374	AG(AcqSpinLock() ⇒ AF RelSpinLock())	Yes	18.8	✓
WDD2	374	AG(AcqSpinLock() ⇒ AF RelSpinLock())	No	14.1	χ
WDD3	58	AF(EnCritRegion() ⇒ EG ExCritRegion())	Yes	12.5	χ
WDD3	58	AF(EnCritRegion() ⇒ EG ExCritRegion())	No	9.6	✓
WDD4	302	AG(added_socket > 0 ⇒ AFEG STATUS_OK)	Yes	30.2	✓
WDD4	302	AG(added_socket > 0 ⇒ AFEG STATUS_OK)	No	72.4	χ
Bakery	37	AG(Noncritical ⇒ AF Critical)	Yes	2.9	✓
Bakery	37	AG(Noncritical ⇒ AF Critical)	No	16.4	χ
Prod-Cons	30	AG($p_i > 0 \Rightarrow$ AF $q_i <= 0$)	Yes	18.5	✓
Prod-Cons	30	AG($p_i > 0 \Rightarrow$ AF $q_i <= 0$)	No	5.5	χ
Chain	48	AG($x \geq 8 \Rightarrow$ AF $x = 0$)	Yes	1.8	✓
Chain	48	AG($x \geq 8 \Rightarrow$ AF $x = 0$)	No	4.7	χ

RECAP

- Introduced the first known method for symbolically proving fair-CTL properties of (infinite-state) integer programs.
- Solution is based on a reduction which allows the use and integrate with any off-the-shelf CTL tool
- Use prophecy variables in the reduction for the purpose of symbolically partitioning fair from unfair executions.
- Implemented as an extension to **T2,** a CTL model checker which returns assertions characterizing the states in which a property holds.