Formal Methods in Software Development

More on Symbolic Model Checking
The μ-calculus

Ivano Salvo

Computer Science Department

SAPIENZA UNIVERSITÀ DI ROMA

Lesson 7, November 16^{th}, 2020

from Lesson 6d:

Symbolic CTL model checking

CTL model checking

The problem is to find three functions such that:

$$check(\mathbf{EX}\ f) = checkEX(check(f))$$

$$check(\mathbf{E}[f\ \mathbf{U}\ g]) = checkEU(check(f), Check(g))$$

$$check(\mathbf{EG}\ f) = checkEG(check(f))$$

Observe that the **parameter of *check* is a CTL formula** φ, its **result is an OBDD** representing the set of states satisfying φ.

The parameters of *checkEX*, *checkEU,* and *checkEG* **are OBDDs**.

CTL model checking

❖ ***checkEX(f(v))*** is strighforward. The OBDD result is equivalent to $\exists v'. f(v') \wedge R(v,v')$.

❖ ***checkEU($f_1(v)$, $f_2(v)$)*** is based on the characterization of **EU** as the least fixpoint of the predicate transformer $\mu Z.\ f_2(v) \vee (f_1(v) \wedge \mathbf{EX}\ Z)$

It is computed a converging sequence of states $Q_1, \ldots, Q_i, \ldots$ Having the OBDD for Q_i and those for $f_1(v)$ and $f_2(v)$ one can easily compute those for Q_{i+1}. Observe that checking $Q_i = Q_{i+1}$ is straighforward (**due to OBDD canonical forms**).

❖ ***checkEG(f(v))*** is based on the characterization of **EG** as the greatest fixpoint of the predicate transformer $\nu Z.\ f_1(v) \wedge \mathbf{EX}\ Z$

Quantified Boolean Formulas

In the previous slide, we used formulas such as: $\exists v'.f(v') \wedge R(v,v')$

They are **quantified boolean formulas**: they are equivalent to propositional formulas, but they allow a more succint representation.

Semantics ($\boldsymbol{\sigma}$ is a variable assignment and $\boldsymbol{\sigma\langle v \leftarrow 0\rangle}$ means that variable v is assigned to 0):

- $\sigma \models \exists v f$ iff $\sigma\langle v \leftarrow 0\rangle \models f$ or $\sigma\langle v \leftarrow 1\rangle \models f$, and
- $\sigma \models \forall v f$ iff $\sigma\langle v \leftarrow 0\rangle \models f$ and $\sigma\langle v \leftarrow 1\rangle \models f$.

They can be represented as OBDD **using restriction**:

- $\exists x f = f|_{x \leftarrow 0} \vee f|_{x \leftarrow 1}$
- $\forall x f = f|_{x \leftarrow 0} \wedge f|_{x \leftarrow 1}$

Lesson 7a:

Symbolic Model Checking with Fairness Constraints

Fairness in Symbolic CTL MC

The goal is to define a procedure *CheckFair* that checks a CTL formula under a set of **fairness constraints** $F = \{P_1, \ldots, P_k\}$ (here we consider only **unconditional fairness**).

This function depends on functions *CheckFairEX, CheckFairEU, CheckFairEG,* fair versions of those already seen.

Symbolic model checking for formulas **EX** *f* and **E** [f_1 **U** f_2] are **similar to the explicit** case. More precisely, let:

$$fair(v) = checkFairEG(\mathbf{EG}\ \mathsf{True})$$

Then:

$$checkFairEX(f(v)) = checkEX(f(v) \wedge fair(v))$$

$$checkFairEX(f(v), g(v)) = checkEX(f(v), g(v) \wedge fair(v))$$

Therefore the problem is again to deal with the problem of computing **EG** *f*. In symbolic model checking, it is again convenient to **model such formula with fixpoints**.

EG f under Fairness Constraints

The set Z of states that satisfies **EG** f given fairness constraints $F = \{P_1, \ldots, P_n\}$ is the largest set satisfying the following properties:

1. all states in Z satisfy f and
2. for all $P_k \in F$ and all states $s \in Z$ there exists a path of states in Z starting at s satisfying P_k.

Therefore, Z must satisfy the following formula:

$$\mathbf{EG}\, f = \upsilon Z.\, f \wedge \bigwedge_{i=1,\ldots,n} \mathbf{EX}\ \mathbf{E}[f\, \mathbf{U}\, (Z \wedge P_i)]$$

This is **not a CTL formula**, since it uses both CTL operators and fixpoints (by contrast, it's a ***μ*-calculus formula**, see later).

First, we show correctness of this formula, by showing that **EG** f is the maximum fixpoint of the equation:

$$Z = f \wedge \bigwedge_{i=1,\ldots,n} \mathbf{EX}\ \mathbf{E}[f\, \mathbf{U}\, (Z \wedge P_i)] \qquad \mathbf{(1)}$$

EG f under Fairness Constraints

Lemma: The fair version of **EG** f is a fixpoint of Eq. (**1**).

Proof: If $s \in$ **EG** f, there exists a fair path starting in s all of whose states satisfy f. Let $s_i \neq s$ such that $s_i \in P_i$. Also s_i is the start of a fair path satisfying **EG** f. By repeatedly apply this argument, it follows that forall i, $s \vDash f \wedge \mathbf{EX\ E}[f\ \mathbf{U}\ (\mathbf{EG}\ f \wedge P_i)]$ and hence $s \vDash f \wedge \bigwedge_{i=1,\ldots,n} \mathbf{EX\ E}[f\ \mathbf{U}\ (Z \wedge P_i)]$.

If s satisfies Eq. (**1**), there exists a finite path to s', such that $s' \vDash \mathbf{EG}\ f \wedge P_i$. Along this path, each state satisfies f and s' is the beginning of a fair path satisfying **EG** f. □

Lemma: The greatest fixpoint of Eq. (**1**) is included in the fair version of **EG** f.

Proof: Let Z be a fixpoint of Eq. (**1**), then $Z \subseteq \mathbf{EG}\ f$. Again, we can build a path $s_1, \ldots, s_n$ in Z such that all states satisfy f and $s_1 \in P_1, \ldots, s_n \in P_n$. The last state is in Z and hence there exists a path back to some state in P_1 etc. So, $Z \subseteq \mathbf{EG}\ f$ and hence **EG** f is the greatest fixpoint. □

Computing fair EG f

From the previous characterization of the fair version of **EG** *f*, the procedure *checkFairEG*(*f*(*v*)) can compute the set of states Sat(**EG** *f*) as:

$$\upsilon Z(v).\, f(v) \wedge \bigwedge_{k=1,\,\dots,\,n} \mathbf{EX}\ \mathbf{E}[f(v)\ \mathbf{U}\ (Z(v) \wedge P_k)]$$

Observe that this implies to compute **several nested fixpoint computations** inside **EU**.

Lesson 7b:

Counterexamples and witnesses

Counterexamples and Witnesses

We remind that the falsification of a formula of the form **AG** f is a path in which at some point $\neg f$ holds (**counterexample**).

Dually, the proof of a formula of the form **EF** f, is a path in which at some point, the formula f holds (**witness**).

The counterexample for a universally quantified formula is the witness for the dual existentially quantified formula.

As usual, we restrict our attention to find witnesses for the three basic CTL opeարators **EX**, **EG**, and **EU**.

A **counterexample** of a formula **AX** f is a path of length 1 s, s' such that $R(s, s')$ and $\mathcal{M}, s \nvDash f$. The **witness** of a formula **EX** f is a path of length 1 s, s' such that $R(s, s')$ and $\mathcal{M}, s \vDash f$. They can be found by inspecting immediate successors of initial states.

A **witness** of a formula **E** $[f$ **U** $g]$ is a path of length k, $s_1, s_2, \ldots s_k$ such that $\mathcal{M}, s_k \vDash g$ and for all $0 \leq j < k$ $\mathcal{M}, s_j \vDash f$. It can be found by a **backward reachability** from Sat(g), therefore during a model checking verification process.

Counterexamples and Witnesses

A **counter-example** of a formula $\mathbf{A}\,[f\,\mathbf{U}\,g]$ can be either:

- ❖ an infinite path $\pi \equiv s_1, s_2, \ldots s_k \ldots$ such that for all k, $\mathcal{M}, s_k \nvDash f \wedge \neg g$, that is $\mathcal{M}, \pi \vDash \mathbf{G}\, f \wedge \neg g$, hence $\mathcal{M}, s \vDash \mathbf{EG}\, f \wedge \neg g$, (witness of $\mathbf{EG}\, f \wedge \neg g$).

- ❖ a finite path $\pi \equiv s_1, s_2, \ldots s_k$ such that for all $0 \leq j < k$ $\mathcal{M}, s_j \vDash f \wedge \neg g$ and $\mathcal{M}, s_k \vDash \neg f \wedge \neg g$ (witness of $\mathbf{E}\,[f \wedge \neg g\ \mathbf{U}\ \neg f \wedge \neg g\,]$).

We are left to deal with witnesses of $\mathbf{EG}\, f$.

Again, for **EG** we will consider the compressed graph of **Strongly Connected Components** of the transition graph of the Kripke structure: this graph does not contain any proper cycle and each infinite path must have a suffix entirely contained in some strongly connected component.

Witnesses for **EG** *f*

Remeber that:

$$(*) \qquad \mathbf{EG}\, f = \nu Z.\, f \wedge \bigwedge_{i=1,\, \dots,\, n} \mathbf{EX}\ \mathbf{E}\, [f\ \mathbf{U}\ (Z \wedge P_i)]$$

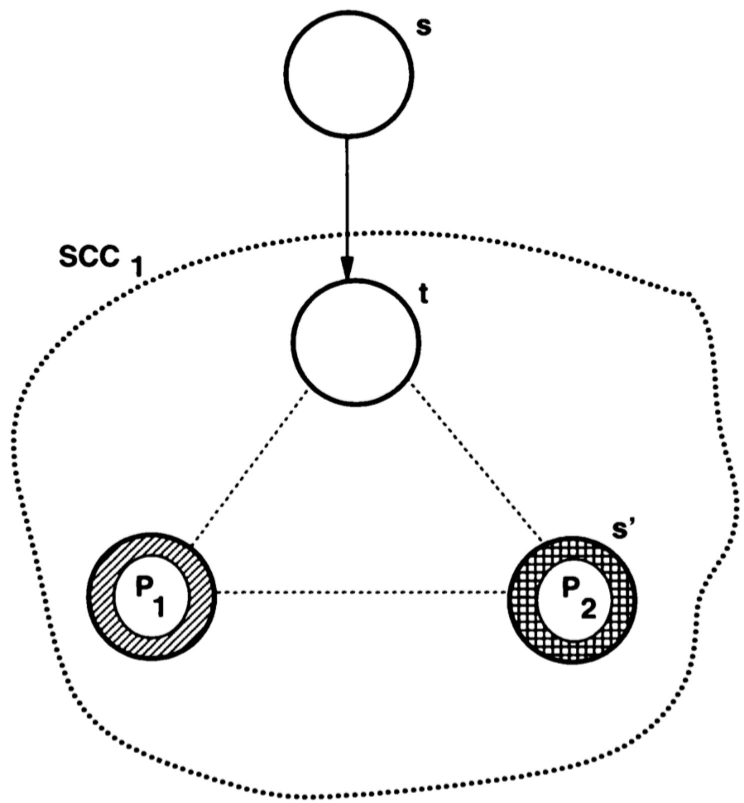
We build a sequence of prefixes of a path π, such that $\pi \models$ **EG** f, until a cycle is found. At each step, we must guarantee that the current prefix can be extended to a fair path satisfying **EG** f.

In the evaluation of $(*)$, **we compute a sequence of fixpoints of the formula E [f U $(Z \wedge P_i)$]**. For each constraint P, we obtain a sequence of sets of states $Q_0^P \subseteq Q_1^P \subseteq Q_2^P \subseteq \dots$ such that Q_k^P is a the set of states in $Z \wedge P$ reachable in i or fewer steps. Therefore **we have for each k and P_i the sequence Q_i^{Pi}**.

Let $s \in$ **EG** f . To minimise the length of the counterexample, we look for the **first fairness constraint that can be reached from s**, looking in Q_0^{Pi} for all P_i in F, then in Q_1^{Pi} and so on.
Since $s \models$ **EG** f, we must eventually find a state t and t has a path of length l to a state u in **EG** $f \wedge P$ and hence it is in **EG** f.
We eliminate P and continue from u…

Witnesses for EG f

At the end we come up with a state s'. We need a path from s' to t to complete a cycle, along states that satisfies f. We need a witness of the formula $\{s'\} \wedge$ **EX E** $[f\ \mathbf{U}\ \{t\}]$. If it is true, we are done (see picture).



Witnesses for EG f

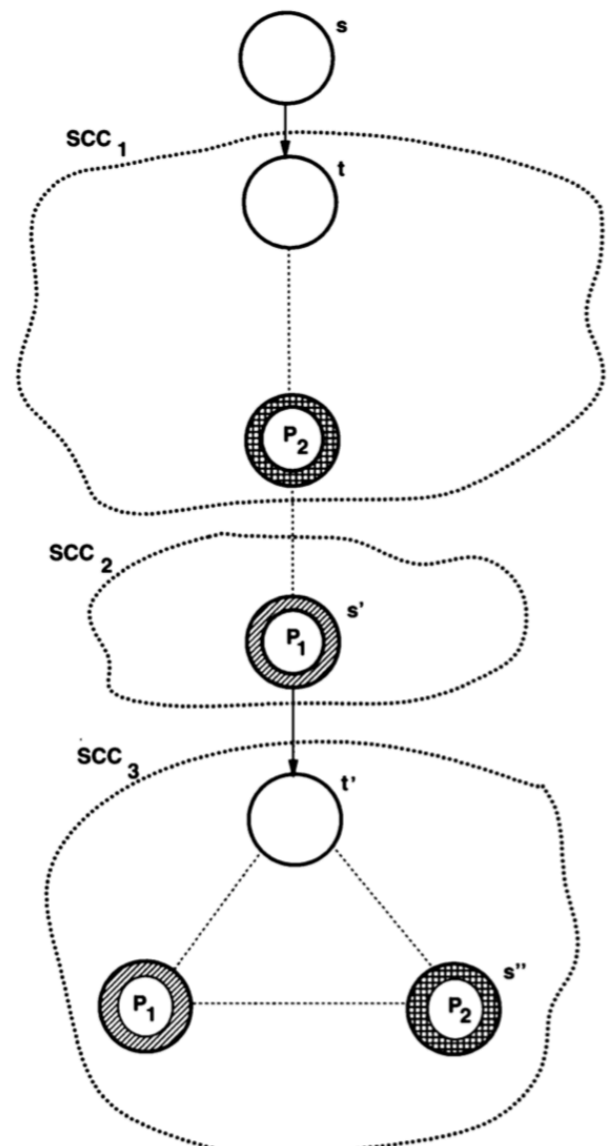
Otherwise, we restart the procedure with fairness constraints F starting from s'.

Since $\{s'\} \wedge$ **EX E** $[f$ **U** $\{t\}]$ is false, s' is not in the SCC of t.

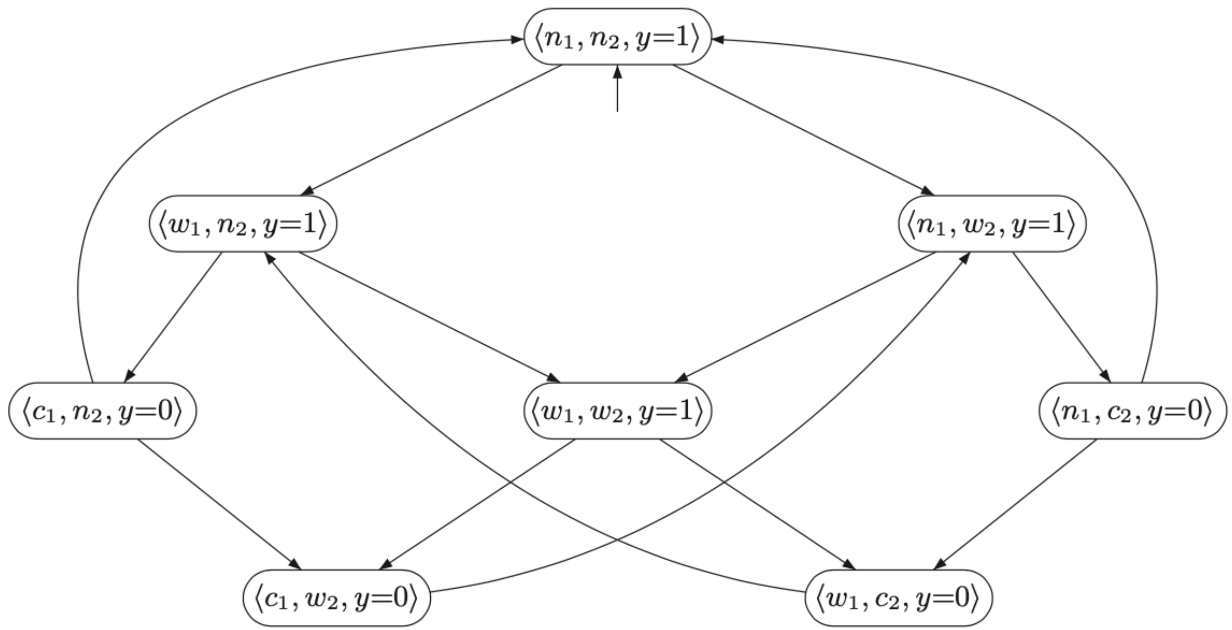
However, $s' \in$ **EG** f and we can continue the process.

Observe that we **descend in the compressed acyclic graph**!

So the process **must terminate**!

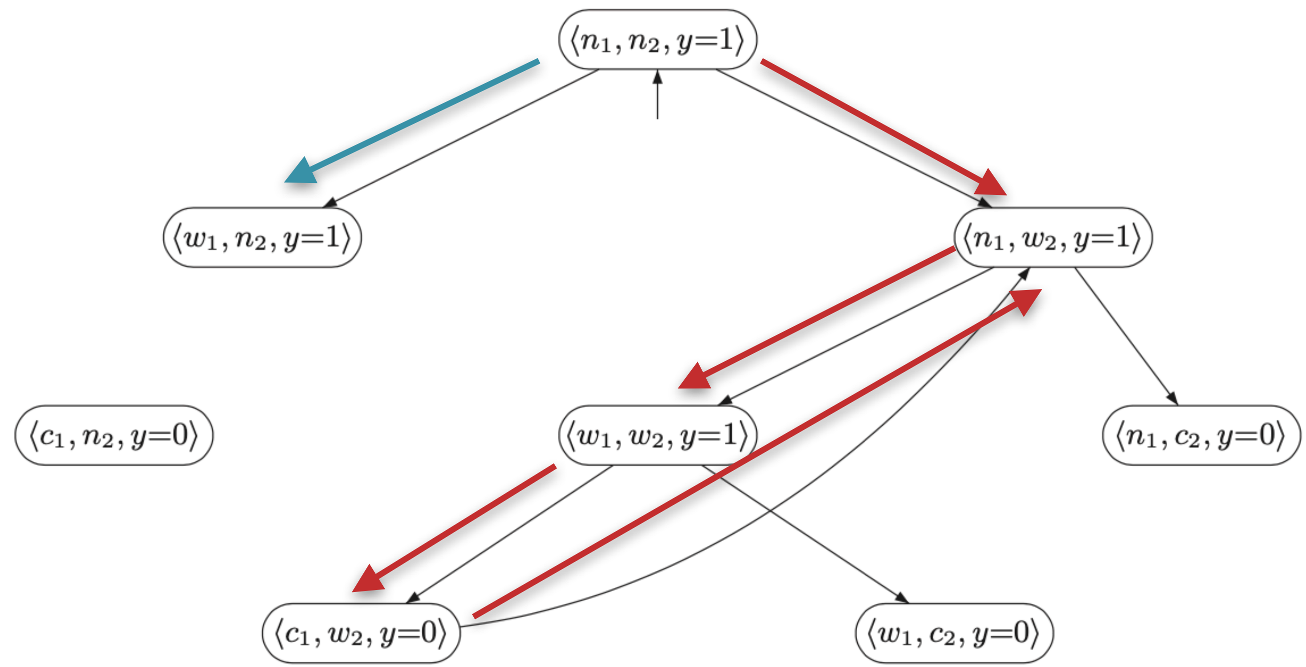


Example: Mutual Exclusion



Let us consider the formula: $\mathbf{A}\ [(n_1 \wedge n_2) \vee w2\ \mathbf{U}\ c_2]$ that states that the process P_2 acquires the critical section once it has started to waiting for it.

Example: Mutual Exclusion



Build the graph for which holds $(n_1 \wedge n_2) \vee w2$ and $\neg\, c_2$.

Let $\pi \equiv \langle n_1, n_2, y{=}1\rangle$ and $\pi' \equiv \langle n_1, w_2, y{=}1\rangle \rightarrow \langle w_1, w_2, y{=}1\rangle \rightarrow \langle c_1, w_2, y{=}0\rangle \rightarrow \langle c_1, w_2, y{=}0\rangle$. Then $\pi\ \pi'^{\omega}$ is a counterexample.

Also $\langle n_1, n_2, y{=}1\rangle \rightarrow \langle w_1, w_2, y{=}1\rangle$ because $\neg\, n_1$ and $\neg w_2$

Lesson 7c:

The μ-calculus

The μ-calculus

Widespread use of **OBDDs** has made **fixpoint-based algorithms appealing** for many applications.

The μ-calculus **explicitely considers fixpoints** in its sintax.

Model-checking procedures follow a **bottom-up approach starting from sub-formulas**. Fixpoints are computed by using iteration and convergence of ascending chains of sets of states.

A naïve approach requires a complexity $\mathcal{O}(n^k)$ to evaluate a μ-calculus formula, where n is the number of states and k is the **nesting** of fixpoint to be evaluated.

More sophisticated algorithms are $\mathcal{O}(n^d)$ where d is the number of alternation greatest/leatest fixpoint.

Syntax of the μ-calculus

Formulas of the μ-calculus are relative to a transition system.

Here we consider a transition system of the form $\mathcal{M}=(S, T, L)$, similar to Kripke structures, but where the **transition relation T is partioned into a family of actions** $\alpha \subseteq S \times S$.

Let us consider a set of relational variables, $Vars=\{Q, Q_1, Q_2, \ldots\}$

- $p \in AP$ **then** p is a formula;
- a relational variable $Q \in Vars$ is a formula;
- if f and g are formulas **then** $f \vee g$, $f \wedge g$, and $\neg f$ are formulas.
- If f is a formula, $\alpha \in T$ **then** $[\alpha] f$ and $\langle\alpha\rangle f$ are formulas.
- If $Q \in Vars$ and f is a formula **then** $\mu Q.\, f$ and $\upsilon Q.\, f$ are formulas

Semantics of the μ-calculus

A formula f is **interpreted** as **the set of states in which f is true**.

We write such set $[\![f]\!]_{\mathcal{M}, e}$ in the transition system $\mathcal{M}$ and in the environment e, where **e is a map from variables to subsets of S**.

$$[\![p]\!]_{\mathcal{M}, e} = \{ s \mid p \in L(s)\} \qquad [\![Q]\!]_{\mathcal{M}, e} = e(Q)$$

$$[\![\neg f]\!]_{\mathcal{M}, e} = S \setminus [\![f]\!]_{\mathcal{M}, e}$$

$$[\![f \vee g]\!]_{\mathcal{M}, e} = [\![f]\!]_{\mathcal{M}, e} \cup [\![g]\!]_{\mathcal{M}, e} \qquad [\![f \wedge g]\!]_{\mathcal{M}, e} = [\![f]\!]_{\mathcal{M}, e} \cap [\![g]\!]_{\mathcal{M}, e}$$

$$[\![[\alpha] f]\!]_{\mathcal{M}, e} = \{s \mid \forall t.\ (s, t) \in \alpha \text{ and } t \in [\![f]\!]_{\mathcal{M}, e}\}$$

$$[\![\langle\alpha\rangle f]\!]_{\mathcal{M}, e} = \{s \mid \exists t.\ (s, t) \in \alpha \text{ and } t \in [\![f]\!]_{\mathcal{M}, e}\}$$

$$[\![\mu Q. f]\!]_{\mathcal{M}, e} = \text{lfp } \tau, \text{ where } \tau(Z) = [\![f]\!]_{\mathcal{M}, e[Q \leftarrow Z]}$$

$$[\![\nu Q. f]\!]_{\mathcal{M}, e} = \text{gfp } \tau, \text{ where } \tau(Z) = [\![f]\!]_{\mathcal{M}, e[Q \leftarrow Z]}$$

Monotonicity

All logical operators, **except negation**, **are monotonic**: $f \to f'$ implies $f \vee g \to f' \vee g$, $f \wedge g \to f' \wedge g$, $[\alpha] f \to [\alpha] f'$, and $\langle\alpha\rangle f \to \langle\alpha\rangle f'$.

Negation must be restricted to atomic propositions.

Using deMorgan's laws and duality, we can always push negation to atomic propositions:

$$\neg[\alpha] f \equiv \langle\alpha\rangle\neg f, \qquad \neg\langle\alpha\rangle f \equiv [\alpha]\neg f,$$
$$\neg\mu Q.\, f \equiv \upsilon Q.\, \neg f(\neg Q), \qquad \neg\upsilon Q.\, f \equiv \mu Q.\, \neg f(\neg Q).$$

Observe that if bound variables are under a **even number of negations**, they **will be negation free** at the end of this process.

Remember that in this finite world:

$$[\![\mu Q.\, f]\!]_{\mathcal{M}, e} = \cup_i\, \tau^i(\text{false}) \quad \text{and} \quad [\![\upsilon Q.\, f]\!]_{\mathcal{M}, e} = \cap_i\, \tau^i(\text{true})$$

Expressivity: CTL and μ-calculus

The μ-calculus is expressive enough to **embody CTL.**

We can easily translate any CTL formula into the μ-calculus , by using **fixpoint characterization** of CTL operators **EG** and **EU**.

$$\mathcal{T}(p) = p \qquad \mathcal{T}(\neg f) = \neg\mathcal{T}(f)$$

$$\mathcal{T}(f \wedge g) = \mathcal{T}(f) \wedge \mathcal{T}(g) \qquad \mathcal{T}(\mathbf{EX}\, f) = \langle\alpha\rangle\, \mathcal{T}(f)$$

$$\mathcal{T}(\mathbf{E}\,[f\,\mathbf{U}\,g]) = \mu Z.\, \mathcal{T}(g) \vee (\mathcal{T}(f) \wedge \langle\alpha\rangle\, Z)$$

$$\mathcal{T}(\mathbf{EG}\, f) = \upsilon Z.\, \mathcal{T}(f) \wedge \langle\alpha\rangle\, Z$$

Example: The CTL formula **EG** (**E** [p **U** q]) is translated into the μ-calculus expression $\upsilon Y.\, (\mu Z.\, (q \vee (p \wedge \langle\alpha\rangle\, Z)) \wedge \langle\alpha\rangle\, Y)$.

Theorem: Let $\mathcal{M}=(S, R, L)$ be a Kripke structure and α be the transition relation R. Let f be a CTL formula. Then, for all $s \in S$:

$$\mathcal{M}, s \models f \quad \Leftrightarrow \quad s \in [\![\mathcal{T}(f)]\!]_{\mathcal{M}}$$

Evaluating fixpoint formulas

There is a naïve recursive algorithm to compute the set of states $[\![f]\!]_{\mathcal{M}, e}$ recursively on the syntactic structure of f:

def $eval(f, e)$:
  if $f \equiv p$ **then return** $\{ s \mid p \in L(s) \}$
  if $f \equiv Q$ **then return** $e(Q)$;
  if $f \equiv g_1 \wedge g_2$ **then**
    return $eval(g_1, e) \cap eval(g_2, e)$;
  if $f \equiv g_1 \vee g_2$ **then**
    return $eval(g_1, e) \cup eval(g_2, e)$;
  if $f \equiv \langle \alpha \rangle g$ **then**
    return $\{ s \mid \exists t\ [s \rightarrow_\alpha t \text{ and } t \in eval(g, e)] \}$;
  if $f \equiv [\alpha]\ g$ **then**
    return $\{ s \mid \forall t\ [s \rightarrow_\alpha t \text{ and } t \in eval(g, e)] \}$;
…

Recursive calls

Evaluating fixpoint formulas

```
…
    if f ≡ μQ.g(Q) then
        Q_val = FALSE
        repeat
            Q_old = Q_val
            Q_val = eval(g, e[Q=Q_val]);
        until Q_old = Q_val
        return Q_val;
    fi
…
```

Least fixpoint computation

Can trigger nested fixpoint computations with different values for variables! $\mathcal{O}(n^k)$, n number of states, k nesting

```
…
    if f ≡ νQ.g(Q) then
        Q_val = TRUE
        repeat
            Q_old = Q_val
            Q_val = eval(g, e[Q=Q_val]);
        until Q_old = Q_val
        return Q_val;
    fi
end
```

Greatest fixpoint computation

Repr. μ-calculus with OBDDs

States are represented by a vector $\boldsymbol{x}$ of boolean variables. As usual, there exists an OBDD, $O_p(\boldsymbol{x})$ for each atomic proposition $p \in AP$. Each transition relation α is an OBBD $O_\alpha(\boldsymbol{x}, \boldsymbol{x}')$.

The function assoc[Q_i] plays the role of environments in OBDD representation and return the OBDD corresponding to the set of states associated to the relational variable Q_i.

We define a function $\mathcal{B}(f,$ assoc$)$ that taking a μ-calculus formula f and an association list assoc that assign an OBDD to each **free relational variables** of f, returns an OBDD corresponding to the semantics of f, that is $[\![f]\!]_{\mathcal{M}, e}$

Repr. μ-calculus with OBDDs

$\mathcal{B}(p, \text{assoc}) = \mathrm{O}_p(\boldsymbol{x})$ $\qquad$ $\mathcal{B}(Q_i, \text{assoc}) = \text{assoc}(Q_i)$

$$\mathcal{B}(\neg f, \text{assoc}) = \neg\, \mathcal{B}(f, \text{assoc})$$

$$\mathcal{B}(f \wedge g, \text{assoc}) = \mathcal{B}(f, \text{assoc}) \wedge \mathcal{B}(g, \text{assoc})$$

$$\mathcal{B}(f \vee g, \text{assoc}) = \mathcal{B}(f, \text{assoc}) \vee \mathcal{B}(g, \text{assoc})$$

$$\mathcal{B}(\langle\alpha\rangle f, \text{assoc}) = \exists \boldsymbol{x}' \, [\mathrm{O}_\alpha(\boldsymbol{x}, \boldsymbol{x}') \wedge \mathcal{B}(f, \text{assoc})(\boldsymbol{x}')]$$

$$\mathcal{B}([\alpha] f, \text{assoc}) = \forall \boldsymbol{x}' \, [\mathrm{O}_\alpha(\boldsymbol{x}, \boldsymbol{x}') \wedge \mathcal{B}(f, \text{assoc})(\boldsymbol{x}')]$$

where $\mathrm{O}(\boldsymbol{x}')$ is the OBDD in which occurrence of each variable $\boldsymbol{x}_i$ is substituted by its primed version $\boldsymbol{x}'_i$.

$$\mathcal{B}(\mu Q.\, f, \text{assoc}) = \text{fix}(f, \text{assoc}, \mathrm{O}_{\text{false}}, Q)$$

$$\mathcal{B}(\nu Q.\, f, \text{assoc}) = \text{fix}(f, \text{assoc}, \mathrm{O}_{\text{true}}, Q)$$

where fix is the OBDD version of usual gfp/lfp iterative computation (see next slide)

Fix computation with OBDDs

```
def fixOBDD(f, assoc, ℬ, Q)
    O_res = ℬ
    repeat
        O_old = O_res
        O_res = ℬ (f, assoc⟨Q= O_old⟩);
    until O_old = O_res
    return Q_val;
```

where assoc$\langle Q{:=}\ O_{old} \rangle$ creates a new variable Q and associate the OBDD O_{old} with Q.

Optimizations

The overall complexity is $\mathcal{O}(|\mathcal{M}| \cdot |f| \cdot |S|^k)$, being $\mathcal{O}(|\mathcal{M}| \cdot |f|)$ the cost of a **single iteration** and $|S|^k$ the maximum number of iteration due to **nested fixpoint computations**.

Observation: it is **not necessary** to **reinitialize from** FALSE (or TRUE) nested least (or greatest) fixpoint computations of the same type of its outermost fixpoint.

This works because of the following corollary of Knarster-Tarski fixpoint theorem:

Corollary: τ monotonic and $W \subseteq \mu\tau$, then $\tau^i(W) \subseteq \mu\tau$.

Definition: The **alternation depth** $\#f$ of f is 0 if $f \equiv p \in AP$, $\max\{\#g, \#h\}$ if $f \equiv g \vee h$, $f \equiv g \wedge h$, $\#g$ if $f \equiv \neg g$ or $f \equiv [\alpha]g$ or $f \equiv \langle\alpha\rangle g$.
The alternation depth of $f \equiv \mu Q.\, g$ is the maximum between 1, $\#g$ and $1+\max\{\#h \mid h \equiv \nu Q.\, h'$ is a top-level subformula of $g\}$.
The alternation depth of $f \equiv \nu Q.\, g$ is the maximum between 1, $\#g$ and $1+\max\{\#h \mid h \equiv \mu Q.\, h'$ is a top-level subformula of $g\}$.

Optimizations: example

Let us consider the formula: $\mu Q_1.g_1(Q_1, \mu Q_2.g_2(Q_1, Q_2))$: for each iteration of the outermost fixpoint, we need to compute the inner fixpoint of the predicate transformer $\tau(Q_1)=\mu Q_2.g_2(Q_1, Q_2)$.

When evaluating the outermost fixpoint, we start with $Q_1^0=$ FALSE and then computing $\tau(Q_1^0)$: this requires iteration of the inner fixpoint, computing a sequence FALSE $=Q_2^{0,0} \subseteq Q_2^{0,1} \subseteq \ldots \subseteq Q_2^{0,\omega}$ and so we get the first approximation $Q_1^1=g_1(Q_1^0, Q_2^{0,\omega})$.

The next inner fixpoint computation of $\tau(Q_1^1)$, will start from $Q_2^{1,0}= Q_2^{0,\omega} =\tau(Q_1^0)$ **rather than** $Q_2^{1,0}=$ FALSE. In general we start the inner fixpoint in the i^{th} iteration of the outer fixpoint, from $Q_2^{i,0}= Q_2^{i\text{-}1,\omega}$.

As a consequence, the computation of n nested leatest fixpoint (or n nested greatest fixpoint) computations can be computed in a number of steps bounded by $n \cdot |S|$, rather than in $|S|^n$ as in the naïve algorithm.

Optimizations: implementation

As a consequence, if the **alternation depth** of a formula f is d, the algorithm can compute fixpoint in $\mathcal{O}((|f| \cdot n)^d)$, because $|f|$ is an upperbound of the number of nested fixpoint **of the same kind.**

The algorithm is similar to the naïve version, **except that it stores intermediate approximations in an array** F_i (see next slide), whose **size is the total number of fixpoint computations** (equivalently, the **number of relational variables**)

When Q_j is bounded by μ (resp. ν), F_j is initialised to FALSE (resp. TRUE) and reset to FALSE (resp. TRUE) only when starts an **outermost** greatest (resp. leatest) **fixpoint computation**.

Optimized fixpoint computation

…
if $f \equiv \mu Q_i.g(Q_i)$ **then**
  forall top-level gfp subf. $\upsilon Q_i.g_i(Q_i)$ **do** F_i=TRUE
  repeat
    $Q = F_i$
    $F_i = eval(g, e[Q = F_i]);$
  until $Q = F_i$
  return F_i
fi
…

reset only greatest fixpoint computations inside the leatest fixpoint computation

Least fixpoint computation

…
if $f \equiv \upsilon Q_i.g(Q_i)$ **then**
  forall top-level gfp subf. $\mu Q_i.g_i(Q_i)$ **do** F_i=FALSE
  repeat
    $Q = F_i$
    $F_i = eval(g, e[Q = F_i]);$
  until $Q = F_i$
  return F_i
fi

reset only leatest fixpoint computations inside the greatest fixpoint computation

Greatest fixpoint computation

Complexity Considerations

The model checking problem for the μ-calculus has been proven to be in ***NP* ∩ co-*NP***.

This essentially comes from the following facts:

1. the problem is in NP because it is polynomial to check **if a given guess for a fixpoint is indeed a fixpoint**.
2. in the μ-calculus **we can easily negate formulas**;

Clarke etal. **conjecture** that **there exists no polynomial algorithm**… but it's very difficult to prove this statement.

In particular, if it would be *NP*-complete, then *NP*=co-*NP* which is unlikely to be true.

Lesson 7c:

Symbolic LTL model checking

Symbolic LTL MC: ideas

We sketch briefly how to adapt LTL model-checking to a symbolic procedure based again on a tableau construction.

The basic idea of LTL symbolic model checking is similar to that of on-the-fly LTL model checking.

We check $\mathcal{M}, s \models \mathbf{E} f$ building a Kripke structure $\mathcal{T}=(S_T, R_T, L_T)$ from the formula f, **to represent all paths that satisfies f.**

Then, we build the product Kripke structure $\mathcal{M} \otimes \mathcal{T}$, and **check on $\mathcal{M} \otimes \mathcal{T}$ if there exists a state such that $s \in \mathrm{Sat}(f)$.**

Elementary Formulas

Given a set of atomic propositions A_f occurring in f, the set of states S_T of the Kripke structure $\mathcal{T}$ represents sets of subformulas of f.

Each state $s \in S_T = \mathcal{P}(el(f))$ is a set of **elementary propositions**.

- $el(p) = \{p\}$ if $p \in AP_f$.
- $el(\neg g) = el(g)$.
- $el(g \vee h) = el(g) \cup el(h)$.
- $el(\mathbf{X}\, g) = \{\mathbf{X}\, g\} \cup el(g)$.
- $el(g\, \mathbf{U}\, h) = \{\mathbf{X}(g\, \mathbf{U}\, h)\} \cup el(g) \cup el(h)$.

The labeling $L_T(s)$ is defined in such a way that each state is labeled with the set of atomic propositions contained in the state.

Building the transition relation

To define the transition relation, we need to define the set of states that satisfies a given formula in $el(f)$ as follows (observe why we don't need to add negations in $el(f)$):

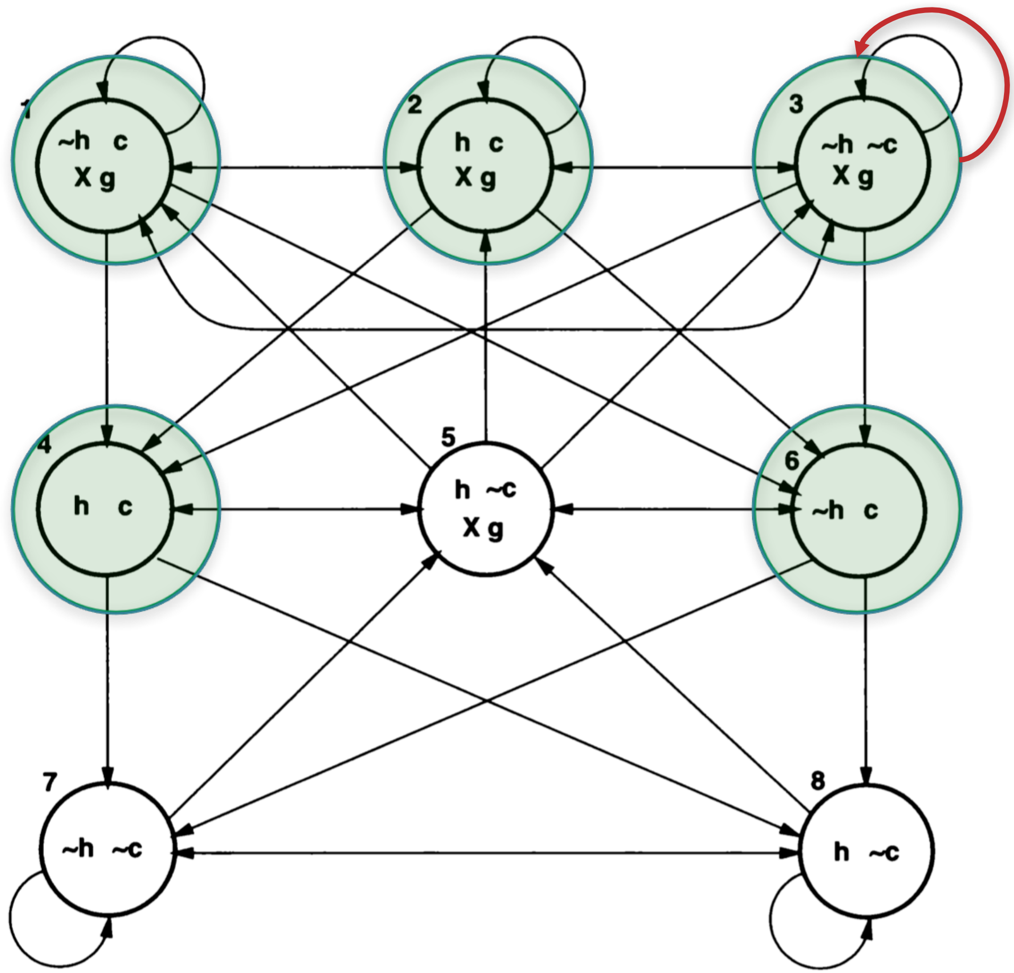
- $sat(g) = \{ s \mid g \in s \}$ where $g \in el(f)$.
- $sat(\neg g) = \{ s \mid s \notin sat(g) \}$.
- $sat(g \vee h) = sat(g) \cup sat(h)$.
- $sat(g \mathbf{U} h) = sat(h) \cup \big(sat(g) \cap sat(\mathbf{X}(g \mathbf{U} h))\big)$.

Again, we want to define R_T in such a way that each formula elementary formula in s in satisfied in s. As usual, we must take care of $\mathbf{X}\, g$ and $\neg\mathbf{X}\, g$.

$$R_T(s, s') = \bigwedge_{\mathbf{X} g \in el(f)} s \in sat(\mathbf{X}\, g) \Leftrightarrow s' \in sat(g).$$

Example

g = ¬heat **U** close. (microwave oven example)



There exists some paths that do not satisfy *g*: for example **the path that loops forever in state 3**, where **close never holds**.

Properties of $\mathcal{T}$

Theorem. Let $\mathcal{T}$ be the tableau for the path formula f. Then, for every Kripke structure $\mathcal{M}$ and every path π' of $\mathcal{M}$, if $\mathcal{M}, \pi' \models f$, then there is a path π in $\mathcal{T}$ such that starts in a state of sat(f) such that labels(π') $|_{APf}$ = labels(π).

Proof: rather technical. Omitted (see Clarke etal.). $\square$

Then, having $\mathcal{T} = (S_T, R_T, L_T)$ and $\mathcal{M} = (S_M, R_M, L_M)$, we build the product $\mathcal{P} = (S, R, L)$ as follows:

- $S = \{ (s, s') \mid s \in S_T, s' \in S_M \text{ and } L_M(s') |_{AP_f} = L_T(s) \}$.
- $R((s, s'), (t, t'))$ iff $R_T(s, t)$ and $R_M(s', t')$.
- $L((s, s')) = L_T(s)$.

R may fail to be total: we remove states without successors.

P contains **exactly** those paths $\pi''=(s_i, s_i')$ such that $L_T(s_i)=L_M(s_i')$

Properties of $\mathcal{T}$

Theorem. $\mathcal{M}, s' \models$ **E** f if and only if there exists $s \in \mathcal{T}$ such that $(s, s') \in \text{sat}(f)$ and $\mathcal{P}, (s, s') \models$ **EG** true under the fairness constraints $\{\text{sat}(\neg(g\ \mathbf{U}\ h) \vee h \mid (g\ \mathbf{U}\ h)$ occurs in $f\}$.

Proof: rather technical. Omitted (see Clarke etal.). $\square$

A path that satisfies the fairness constraint $\{\ \text{sat}(\neg(g\ \mathbf{U}\ h) \vee h \mid (g\ \mathbf{U}\ h)$ occurs in $f\}$ has the property that no subformula of the form $(g\ \mathbf{U}\ h)$ holds almost always on a path while h remain false.

Formula **EG** true under fairness constraints can be checked by using CTL (symbolic) model checking.

LTL Symbolic Model Checking

Representation of $\mathcal{T}$: associate to each formula g in $el(f)$ a boolean variable v_g. $\mathcal{M}$ and $\mathcal{T}$ can be defined over variables in AP_f and some additional variable for formulas in $el(f)$.

States in $\mathcal{M}$ has the shape (p, q), with p boolean variables for atomic proposition AP_f and q variables that are not mentioned in f.

States in $\mathcal{T}$ has the shape (p, r) with r variables of non atomic formulas in the tableau of f.

As usual, transition relations are predicates over two copies, v and v' of state variables. In particular, $\mathcal{P} = \mathcal{M} \otimes \mathcal{T}$, we have:

$$R_P(p, q, r, p', q', r') = R_T(p, r, p', r') \wedge R_M(p, q, p', q')$$

On this Kripke structure, we can use **CTL model checking** with fairness constraints to determine a set of states V=**EG** true holds. Moreover, we have that $\mathcal{M}, s \models \mathbf{E} f$ if and only if s is represented by (p, q) and $\exists r.\ (p, q, r) \in V$ and $(p, r) \in \mathrm{sat}(f)$.

That's all Folks!

Thanks for your attention…

…Questions?