Formal Methods in Software Development

Ivano Salvo

and Igor Melatti

Computer Science Department

SAPIENZA
UNIVERSITÀ DI ROMA

Lesson **8**, November 12th, 2019

Lesson 8a:

Symbolic Model Checking with Fairness Constraints

Fairness in Symbolic CTL MC

We have to find 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 of the past lesson.

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 si 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 $F = \{P_1, \ldots, P_k\}$ 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 sequence of states in Z starting at *s* satisfying P_k.

Therefore, Z must satisfy the following formula:

$$\mathbf{EG}\ f = \nu Z.\ f \wedge \bigwedge_{k=1,\ldots,n} \mathbf{EX}\ \mathbf{E}[f\ \mathbf{U}\ (Z \wedge P_k)]$$

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 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*, $s \models f \wedge$ **EX E**[*f* **U** (**EG** $f \wedge P_i$)] and hence $s \models f \wedge \bigwedge_{i=1, \ldots, n}$ **EX E**[*f* **U** ($Z \wedge P_i$)].

If *s* satisfies Eq. (**1**), there exists a finite path to *s'*, such that $s' \models$ **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 $Z \subseteq$ **EG** *f*. Again, we can build a path $s_1, \ldots, s_k$ in Z such that all states satisfy *f* and $s_1 \in P_1, \ldots, s_k \in P_k$. The last state is in Z and hence there exists a path back to some path in P_1 etc. So, $Z \subseteq$ **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:

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

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

Lesson 8b:

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 ¬*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**).

Therefore, the counterexample for a universal 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 opearators **EX**, **EG**, and **EU**.

Again 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:

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

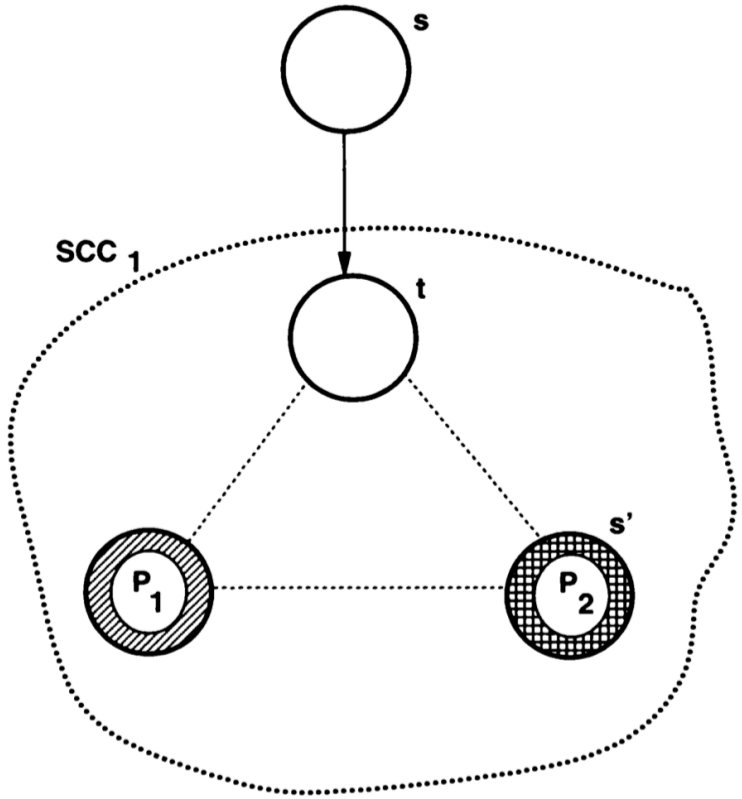
We build a sequence of prefixes of a path π, such that $\pi \models \mathbf{EG}\, f$, until a cycle is found. At each step, we must guarantee that the current prefix can be extended to a fair path satisfying **EG** *f*.

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

Let $s \in \mathbf{EG}\, f$. To minimise the length of the counterexample, we look for the first fairness constraint that can be reached from s, looking in Q_0^P for all P in F, then in Q_1^P and so on.
Since $s \models \mathbf{EG}\, f$, we must eventually find such t. Moreover, t has a path of length i to a state u in $\mathbf{EG}\, f \wedge P$ 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 $\{s'\} \wedge \mathbf{EX}\ \mathbf{E}\ [f\ \mathbf{U}\ \{t\}]$. If it is true, we are done (see picture).



Witnesses for EG f

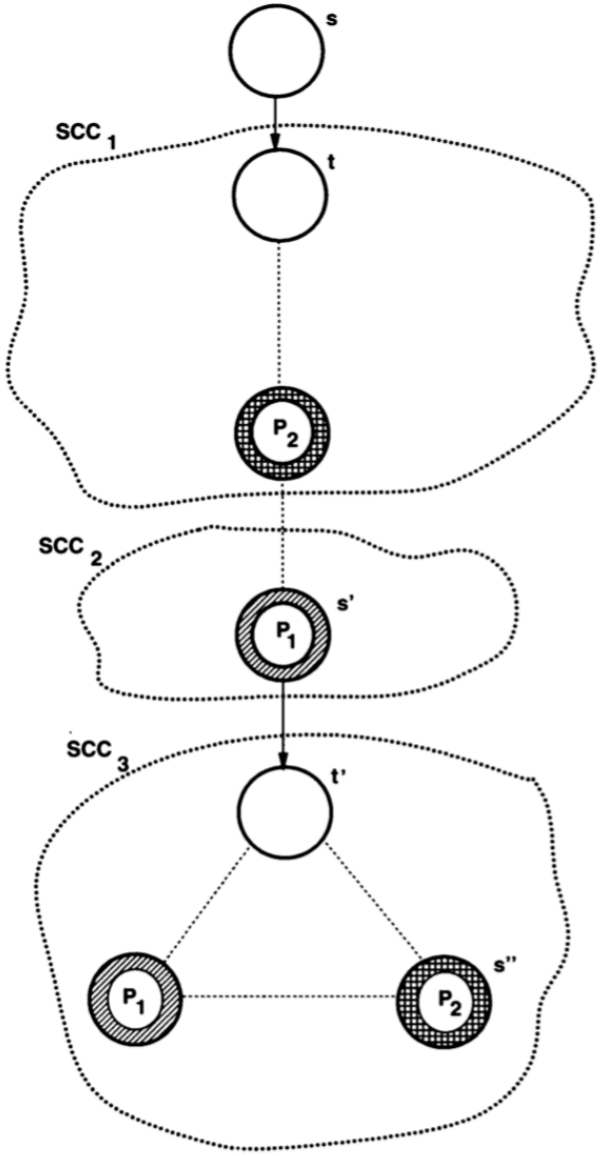
Otherwise, we restart the procedure with fairness constraints F starting from s'.

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**!



Lesson 8c:

Symbolic LTL model checking

Symbolic LTL MC: ideas

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, in order 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 $s \in \text{sat}(f)$.

Elementary Formulas

The Kripke structure $\mathcal{T}$ is on the set of atomic proposition AP_f of atomic propositions occurring as sub-formulas 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 so 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 teh set of states that satisfies a given formula in $el(f)$ as follows (observe why we don't need to add negations in $el(f)$):

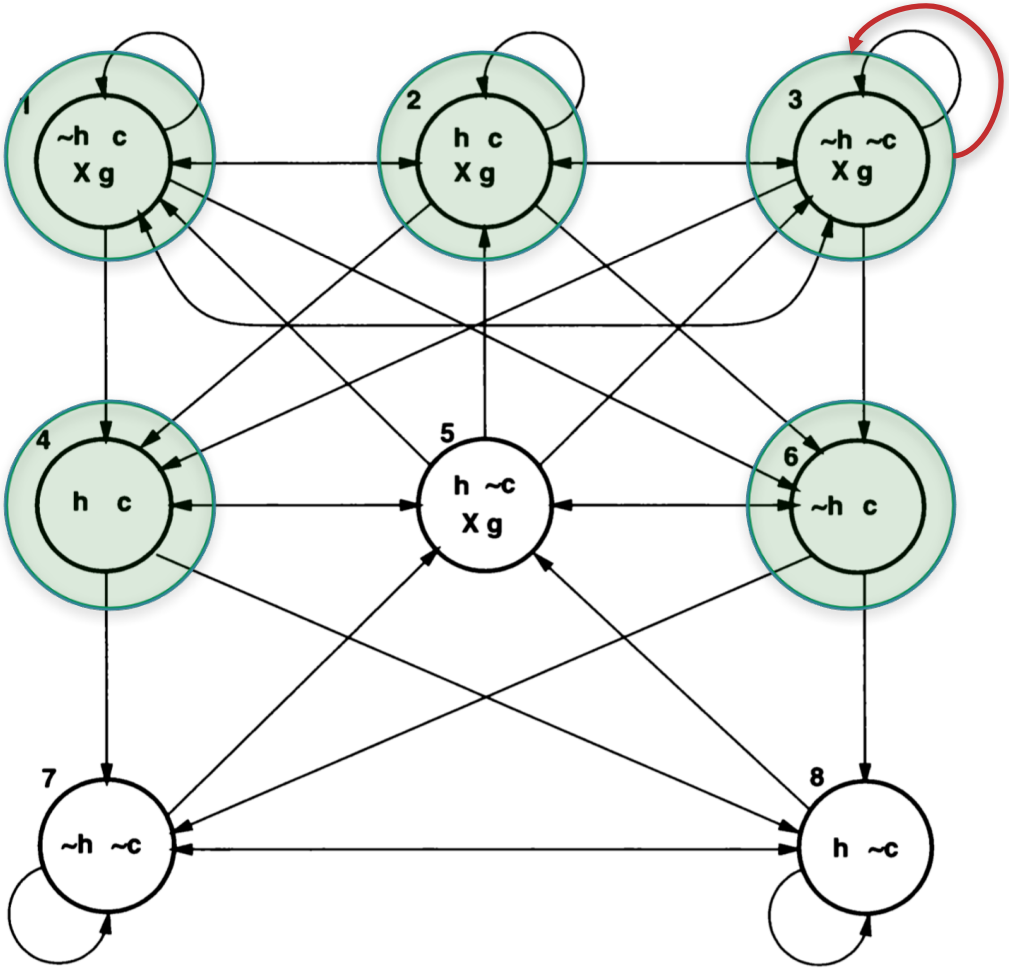
- $sat(g) = \{ s \mid g \in s \}$ where $g \in el(f)$.
- $sat(\neg g) = \{ s \mid s \notin sat(g) \}$.
- $sat(g \vee h) = sat(g) \cup sat(h)$.
- $sat(g \mathbf{U} h) = sat(h) \cup \big(sat(g) \cap sat(\mathbf{X}(g \mathbf{U} h))\big)$.

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

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



There exists some paths that do not satisfy f: 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 $\mathsf{labels}(\pi')|_{APf} = \mathsf{labels}(\pi)$.

Proof: rather technical. Omitted (see Clarke etal.). □

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. $\mathcal{M}, s' \models \mathbf{E}\, f$ if and only if there exists $s \in \mathcal{T}$ such that $(s, s') \in \text{sat}(f)$ and $\mathcal{P}, (s, s') \models \mathbf{EG}$ 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.). □

A path that satisfies the fairness constraint $\{$ 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:

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

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 $(p, r) \in \text{sat}(f)$.

Lesson 8d:

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.

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

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

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

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

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

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

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

Monotonicity

Negation must be restricted to atomic propositions.

Each logical operator, **except negation**, **is 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 $\langle \alpha \rangle f \to \langle \alpha \rangle f'$.

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 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:

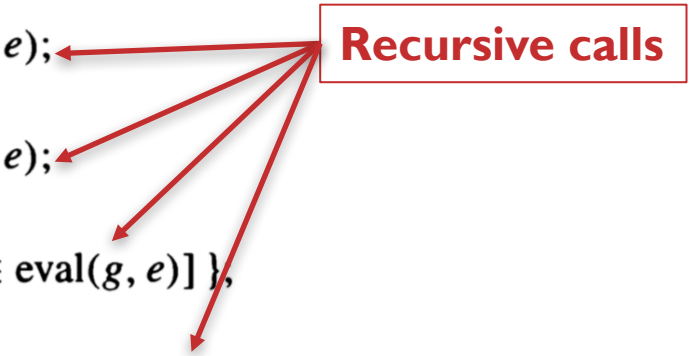
$$[\![\mu Q.\, f]\!]_{\mathcal{M}, e} = \cup_i \tau^i(\mathsf{false}) \quad \text{and} \quad [\![\upsilon Q.\, f]\!]_{\mathcal{M}, e} = \cap_i \tau^i(\mathsf{true})$$

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:

```
1   function eval(f, e)

2   if f = p then return {s | p ∈ L(s)};
3   if f = Q then return e(Q);
4   if f = g1 ∧ g2 then
5         return eval(g1, e) ∩ eval(g2, e);
6   if f = g1 ∨ g2 then
7         return eval(g1, e) ∪ eval(g2, e);
8   if f = ⟨a⟩g then
9         return { s | ∃t [s -a-> t and t ∈ eval(g, e)] };
10  if f = [a]g then
11        return { s | ∀t [s -a-> t implies t ∈ eval(g, e)] };
```



Evaluating fixpoint formulas

```
12  if f = μQ.g(Q) then
13      Q_val := False;
14      repeat
15          Q_old := Q_val;
16          Q_val := eval(g, e [Q ← Q_val]);
17      until Q_val = Q_old;
18      return Q_val;
19  end if;
```

Least fixpoint computation

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

```
20  if f = νQ.g(Q) then
21      Q_val := True;
22      repeat
23          Q_old := Q_val;
24          Q_val := eval(g, e [Q ← Q_val]);
25      until Q_val = Q_old;
26      return Q_val;
27  end if;
28  end function
```

Greatest fixpoint computation

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.

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.

Example: Let us consider the formula: $\mu Q_1.g_1(Q_1, \mu Q_2.g_2(Q_1, Q_2))$. Let $\tau(Q_1)=\mu Q_2.g_2(Q_1, Q_2)$ be a monotonic predicate transformer. When evaluating the outermost fixpoint, we start with Q_1^0=false and then computing $\tau(Q_1^0)$: this is done by iteration of the inner fixpoint, computing a sequence $Q_2^{0,0} \subseteq Q_2^{0,1} \subseteq \ldots \subseteq Q_2^{0,\omega}$ 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 $Q_2^{i,0}= Q_2^{i\text{-}1,\omega}$.

Optimizations

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

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

Since we never restart the inner computations, we need $\mathcal{O}(kn)$ instead of $\mathcal{O}(n^k)$.

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 it stores intermediate approximations in an array (see next slide), whose **size is the total number of fixpoint computations**.

Optimized fixpoint computation

```
12  if f = μQ_i.g(Q_i) then
13      forall top-level greatest fixpoint subformulas νQ_j.g'(Q_j) of g
14          do A[j] := True;
15      repeat
16          Q_old := A[i];
17          A[i] := eval(g, e [Q_i ← A[i]]);
18      until A[i] = Q_old;
19      return A[i];
20  end if;
```

Least fixpoint computation

reset only greatest fixpoint computations.

Greatest fixpoint computation

```
21  if f = νQ_i.g(Q_i) then
22      forall top-level least fixpoint subformulas μQ_j.g'(Q_j) of g
23          do A[j] := False;
24      repeat
25          Q_old := A[i];
26          A[i] := eval(g, e [Q_i ← A[i]]);
27      until A[i] = Q_old;
28      return A[i];
```

reset only least fixpoint computations.

Repr. μ-calculus with OBDDs

States are represented by a vector x of boolean variables.

As usual, there exists an OBDD, $O_p(x)$ for each atomic proposition $p \in AP$.

Each transition relation α is an OBBD $O_\alpha(x, x')$.

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 will 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

$\mathcal{B}(p, \text{assoc}) = O_p(x)$ $\qquad$ $\mathcal{B}(Q_i, \text{assoc}) = \text{assoc}(Q_i)$

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

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

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

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

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

where $O(x')$ is the OBDD in which occurrence of each variable x_i is substituted by its primed version x'_i.

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

$\mathcal{B}(\upsilon Q.\, f, \text{assoc}) = \text{fix}(f, \text{assoc}, O_{\text{true}}, Q)$

where fix is the OBDD version of usual gfp/lfp iterative computation

Fix computation with OBDDs

function fix(f, assoc, $\mathcal{B}$, Q)

$O_{res} = \mathcal{B}$;

repeat

$O_{old} = O_{res}$

$O_{res} = \mathcal{B}\ (f,\ \text{assoc}\langle Q := O_{old} \rangle)$

until $O_{old} == O_{res}$

return $O_{old} == O_{res}$

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

Translating CTL into μ-calculus

$$\mathcal{T}(p) = p$$

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

$$\mathcal{T}(f \wedge g) = \mathcal{T}(f) \wedge \mathcal{T}(g)$$

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

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

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

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

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

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

Complexity Considerations

The model checking problem for the μ-calculus has been proven to be in $\boldsymbol{NP} \cap \textbf{co-}\boldsymbol{NP}$. This essentially comes from the following facts:

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

Clarke etal. conjectures that there exists no polynomial algorithm… but it's very difficult to prove this statement. On the other hand, if it would be *NP*-complete, then *NP*=co-*NP* which is unlikely to be true.

Lesson 8

That's all Folks…

…Questions?