# Model Checking $\mu$-Calculus

## (Based on [Clarke et al. 1999])

Yih-Kuen Tsay
(with help from Kai-Fu Tang and Jinn-Shu Chang)

Dept. of Information Management
National Taiwan University

# Outline

- Introduction
- The Propositional $\mu$-Calculus
- Evaluating Fixpoint Formulae
- Representing $\mu$-Calculus Formulae Using OBDDs
- Translating CTL into the $\mu$-Calculus
- Complexity Considerations

# Introduction

- The propositional $\mu$-calculus is a powerful language for expressing properties of transition systems by using least and greatest fixpoint operators.
- It has gained much attention for two reasons:
  - Many temporal and program logics can be encoded into the $\mu$-calculus.
  - There exist efficient model checking algorithms for this formalism.
- Widespread use of BDDs made fixpoint-based algorithms even more important.

# Introduction (cont.)

- Model checking algorithms for the $\mu$-calculus fall into two categories:
  - Local procedures:
    - for proving that *a specific state* satisfies the given formula
    - not having been combined with BDDs
  - Global procedures:
    - for proving that *all states* in a set satisfy the given formula
    - those based on BDDs prove to be very efficient in practice
- Here, we consider only global model checking.

# Extended Kripke Structures

- Formulae in the $\mu$-calculus are interpreted relative to a transition system.
- To distinguish between different transitions in a system, we modify the definition of a Kripke structure slightly.
- An *extended Kripke structure* $M$ over $AP$ is a tuple $\langle S, T, L \rangle$:
  - $S$ is a nonempty set of states,
  - $T$ is a set of transition relations, and
  - $L : S \to 2^{AP}$ gives the set of atomic propositions true in a state.
- We will refer to each $a \in T$, $a \subseteq S \times S$, as a *transition* (instead of a transition relation).

# $\mu$-Calculus: Syntax

- Let $VAR = \{Q, Q_1, Q_2, \ldots\}$ be a set of *relational variables* (representing unary predicates).
- Each relational variable $Q \in VAR$ can be assigned a subset of $S$.
- The $\mu$-calculus formulae are constructed as follows:
  - If $p \in AP$, then $p$ is a formula.
  - A relational variable is a formula.
  - If $f$ and $g$ are formulae, then $\neg f, f \wedge g, f \vee g$ are formulae.
  - If $f$ is a formula and $a \in T$, then $\langle a \rangle f$ and $[a]f$ are formulae.
  - If $Q \in VAR$ and $f$ is a *syntactically monotone* formula in $Q$, then $\mu Q.f$ and $\nu Q.f$ are formulae.

# Syntactically Monotone Formulae

- A formula $f$ is *syntactically monotone* in $Q$ if all occurrences of $Q$ within $f$ fall under an even number of negations in $f$.
- Consider these formulae:

$$\begin{aligned} f_1 &= \neg((p \vee \neg Q_1) \wedge \neg\langle a\rangle Q_1) \\ f_2 &= (Q_1 \wedge \langle a\rangle Q_1) \vee \neg(p \wedge [a]Q_2) \end{aligned}$$

- $f_1$ is syntactically monotone in $Q_1$.
- $f_2$ is syntactically monotone in $Q_1$, but not syntactically monotone in $Q_2$.

# Intuitive Meaning of $\mu$-Calculus Formulae

- The formula $\langle a \rangle f$ means that $f$ holds in at least one state reachable in one step by making an $a$-transition.
- The formula $[a]f$ means that $f$ holds in all states reachable in one step by making an $a$-transition.
- The formula $\mu Q.f(Q)$ expresses the least fixpoint of $f$.
- The formula $\nu Q.f(Q)$ expresses the greatest fixpoint of $f$.
- The fixpoint operator behaves like a quantifier in first-order logic.
- Variables can be *free* or *bound* by a fixpoint operator.
- We write $f(Q_1, Q_2, \ldots, Q_n)$ to emphasize that a formula $f$ contains free relational variables $Q_1, Q_2, \ldots, Q_n$.

# $\mu$-Calculus: Semantics

- We write $s \xrightarrow{a} s'$ to mean $(s, s') \in a$.
- The *environment* $e : VAR \rightarrow 2^S$ is an interpretation for free variables.
- We denote by $e[Q \leftarrow W]$ a new environment that is the same as $e$ except that $e[Q \leftarrow W](Q) = W$.
- A formula $f$ is interpreted as a set of states in which $f$ is true, denoted $[\![f]\!]_M e$, where
  - $M$ is a transition system and
  - $e$ is an environment.

# $\mu$-Calculus: Semantics (cont.)

- $[\![p]\!]_M e = \{s \mid p \in L(s)\}$
- $[\![Q]\!]_M e = e(Q)$
- $[\![\neg f]\!]_M e = S \setminus [\![f]\!]_M e$
- $[\![f \wedge g]\!]_M e = [\![f]\!]_M e \cap [\![g]\!]_M e$
- $[\![f \vee g]\!]_M e = [\![f]\!]_M e \cup [\![g]\!]_M e$
- $[\![\langle a \rangle f]\!]_M e = \{s \mid \exists t[s \xrightarrow{a} t \text{ and } t \in [\![f]\!]_M e]\}$
- $[\![[a] f]\!]_M e = \{s \mid \forall t[s \xrightarrow{a} t \text{ implies } t \in [\![f]\!]_M e]\}$
- $[\![\mu Q.f]\!]_M e$ is the least fixpoint of the predicate transformer $\tau : 2^S \to 2^S$, where $\tau(W) = [\![f]\!]_M e[Q \leftarrow W]$
- $[\![\nu Q.f]\!]_M e$ is the greatest fixpoint of the predicate transformer $\tau : 2^S \to 2^S$, where $\tau(W) = [\![f]\!]_M e[Q \leftarrow W]$

# An Example

Let $f = p \wedge [a]Q$. Formula $f$ defines a predicate transformer $\tau$ as follows.

$$
\begin{aligned}
\tau(W) &= [\![f]\!]_M e[Q \leftarrow W] \\
&= [\![p \wedge [a]Q]\!]_M e[Q \leftarrow W] \\
&= [\![p]\!]_M e[Q \leftarrow W] \cap [\![[a]Q]\!]_M e[Q \leftarrow W] \\
&= \{s \mid p \in L(s)\} \cap \\
&\quad \{s \mid \forall t(s \xrightarrow{a} t \text{ implies } t \in [\![Q]\!]_M e[Q \leftarrow W])\} \\
&= \{s \mid p \in L(s)\} \cap \\
&\quad \{s \mid \forall t(s \xrightarrow{a} t \text{ implies } t \in W)\}
\end{aligned}
$$

# A CTL Formula in $\mu$-Calculus

- Consider **EG** $f$ with fairness constraint $k$.
- Recall that this property can be expressed as a fixpoint:

$$\nu Z \ . \ f \wedge \mathbf{EX}\ \mathbf{E}[f\ \mathbf{U}\ (Z \wedge k)].$$

- Using the fixpoint characterization of **EU**, we obtain

$$\mathbf{E}[f\ \mathbf{U}\ (Z \wedge k)] = \mu Y \ . \ (Z \wedge k) \vee (f \wedge \mathbf{EX}\ Y).$$

- Substituting the right-hand side of the second formula in the first one gives

$$\nu Z \ . \ f \wedge \mathbf{EX}\ (\mu Y \ . \ (Z \wedge k) \vee (f \wedge \mathbf{EX}\ Y)).$$

# A CTL Formula in $\mu$-Calculus (cont.)

- Suppose the system under consideration has just one transition $a$.
- Replace **EX** by $\langle a \rangle$, we obtain the $\mu$-calculus formula

$$\nu Z \;.\; f \wedge \langle a \rangle (\mu Y \;.\; (Z \wedge k) \vee (f \wedge \langle a \rangle Y)).$$

# Negation and Monotonicity

- All negations can be pushed down to the atomic propositions:

$$\begin{aligned} \neg[a]f &\equiv \langle a\rangle \neg f \\ \neg\langle a\rangle f &\equiv [a]\neg f \\ \neg\mu Q.f(Q) &\equiv \nu Q.\neg f(\neg Q) \\ \neg\nu Q.f(Q) &\equiv \mu Q.\neg f(\neg Q) \end{aligned}$$

- Every logical connective except negation is monotonic.
- Bound variables are under an even number of negations, thus they can be made negation-free.
- Therefore, each possible formula in a fixpoint operator is monotonic.
- This ensures the existence of the fixpoints.

# Fixpoint Reviewed

- Let $\tau : 2^S \rightarrow 2^S$ be a monotonic function.
- If $S$ is finite and $\tau$ is monotonic, then $\tau$ is also $\cup$-continous and $\cap$-continuous.
- $\mu Q.\tau(Q) = \bigcup_i \tau^i(False)$, i.e., $\mu Q.\tau(Q)$ is the union of the following ascending chain of approximations:

$$False \subseteq \tau(False) \subseteq \tau^2(False) \subseteq \cdots \subseteq \tau^n(False) \subseteq \cdots$$

- $\nu Q.\tau(Q) = \bigcap_i \tau^i(True)$, i.e., $\nu Q.\tau(Q)$ is the intersection of the following descending chain of approximations:

$$True \supseteq \tau(True) \supseteq \tau^2(True) \supseteq \cdots \supseteq \tau^n(True) \supseteq \cdots$$

# Naive Algorithm

```
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))};
12  if f = μQ.g(Q) then return Lfp(g, e, Q);
13  if f = νQ.g(Q) then return Gfp(g, e, Q);
14  end function
```

# Naive Least Fixpoint Procedure

```
1  function Lfp(g, e, Q)

2  Q_val ← False;
3  repeat
4        Q_old ← Q_val;
5        Q_val ← Eval(g, e[Q ← Q_val]);
6  until Q_val = Q_old;
7  return Q_val;
8  end function
```

# Naive Greatest Fixpoint Procedure

```
1  function Gfp(g, e, Q)

2  Q_val ← True;
3  repeat
4        Q_old ← Q_val;
5        Q_val ← Eval(g, e[Q ← Q_val]);
6  until Q_val = Q_old
7  return Q_val;
8  end function
```

# A Run Sketch

- Consider the calculation of $\mu Q_1.g_1(Q_1, \mu Q_2.g_2(Q_1, Q_2))$.
- We start with the initial approximation $Q_1^0 = \mathit{False}$.
  - Compute the inner fixpoint starting from $Q_2^{00} = \mathit{False}$ until we reach the fixpoint $Q_2^{0\omega}$.
- $Q_1$ is increased to $Q_1^1 = g_1(Q_1^0, Q_2^{0\omega})$.
  - Compute the inner fixpoint starting from $Q_2^{10} = \mathit{False}$ until we reach the fixpoint $Q_2^{1\omega}$.
- $Q_1$ is increased to $Q_1^2 = g_1(Q_1^1, Q_2^{1\omega})$.
- ...
- This continues until we reach the fixpoint $Q_1^\omega$.

# A Run Sketch (cont.)

- Summary of the calculation of $\mu Q_1.g_1(Q_1, \mu Q_2.g_2(Q_1, Q_2))$:

| | | | |
|---|---|---|---|
| $Q_1^0$ | $Q_2^{00}$ | $Q_2^{01}$ | $\cdots$ $Q_2^{0\omega}$ |
| $= \mathit{False}$ | $= \mathit{False};$ | $= g_2(Q_1^0, Q_2^{00});$ | |
| $Q_1^1$ | $Q_2^{10}$ | $Q_2^{11}$ | $\cdots$ $Q_2^{1\omega}$ |
| $= g_1(Q_1^0, Q_2^{0\omega})$ | $= \mathit{False};$ | $= g_2(Q_1^1, Q_2^{10});$ | |
| $\vdots$ | $\vdots$ | | |
| $Q_1^{\omega-2}$ | $Q_2^{(\omega-2)0}$ | $Q_2^{(\omega-2)1}$ | $\cdots$ $Q_2^{(\omega-2)\omega}$ |
| $= g_1(Q_1^{\omega-3}, Q_2^{(\omega-3)\omega})$ | $= \mathit{False};$ | $= g_2(Q_1^{\omega-2}, Q_2^{(\omega-2)0});$ | |
| $Q_1^{\omega-1}$ | $Q_2^{(\omega-1)0}$ | $Q_2^{(\omega-1)1}$ | $\cdots$ $Q_2^{(\omega-1)\omega}$ |
| $= g_1(Q_1^{\omega-2}, Q_2^{(\omega-2)\omega})$ | $= \mathit{False};$ | $= g_2(Q_1^{\omega-1}, Q_2^{(\omega-1)0});$ | |
| $Q_1^{\omega}$ | | | |
| $= g_1(Q_1^{\omega-1}, Q_2^{(\omega-1)\omega})$ | | | |

# Complexity Analysis

- Let $k$ be the maximum nesting depth of fixpoint operators.
- The naive algorithm runs in $O(|M| \cdot |f| \cdot n^k)$ time, where $M$ is the Kripke structure and $n$ the number of states.
  - The innermost fixpoint will be evaluated $O(n^k)$ times.
  - Each individual iteration takes $O(|M| \cdot |f|)$ steps.

# Alternation Depth

- Top-level $\nu$-subformula of $f$: a subformula $\nu Q.g$ that is not contained within any other greatest fixpoint subformula of $f$.
- The top-level $\mu$-subformula of $f$ is defined analogously.
- The *alternation depth* of a formula $f$ is the number of alternations in the nesting of least and greatest fixpoints in $f$, denoted $d(f)$:
  - $d(p) = d(Q) = 0$
  - $d(f \wedge g) = d(f \vee g) = \max(d(f), d(g))$
  - $d(\langle a \rangle f) = d([a]f) = d(f)$
  - $d(\mu Q.f) = \max(1, d(f), 1 + \max(\{d(g) \mid g \text{ is a top level } \nu\text{-subformula of } f\}))$
  - $d(\nu Q.f) = \max(1, d(f), 1 + \max(\{d(g) \mid g \text{ is a top level } \mu\text{-subformula of } f\}))$

# Alternation Depth (cont.)

- Examples:
  - $d(\mu Q.p \vee \langle a \rangle Q) = 1$
  - $d(\nu Q.(q \wedge (p \vee [a]Q)) = 1$
  - $d(\nu Q_1.(\nu Q_2.(p \wedge [a]Q_2) \wedge \langle a \rangle Q_1)) = 1$
  - $d(\nu Q_1.(\mu Q_2.(p \vee \langle a \rangle Q_2) \wedge \langle a \rangle Q_1)) = 2$
- Recall that, for a system with a single transition $a$ and fairness constraint $k$, the $\mu$-calculus formula corresponding to **EG** $f$ is

$$\nu Z \ . \ f \wedge \langle a \rangle (\mu Y \ . \ (Z \wedge k) \vee (f \wedge \langle a \rangle Y)).$$

This formula has an alternation depth of two.

# A Better Algorithm

- An algorithm by Emerson and Lei demonstrates that the value of a fixpoint formula can be computed with $O((|f| \cdot n)^d)$ iterations, where $d$ is the alternation depth of $f$.
- The basic idea exploits sequences of fixpoints that have the same type to reduce the complexity of the algorithm.
- It is unnecessary to re-initialize computations of inner fixpoints with *False* or *True*.
- Instead, to compute a least fixpoint, it is enough to start iterating with any approximation known to be below the fixpoint.

# Lemma 22

- Let $\tau : 2^S \rightarrow 2^S$ be monotonic and $S$ be finite.
- If $W \subseteq \bigcup_i \tau^i(False)$, then $\bigcup_i \tau^i(W) = \bigcup_i \tau^i(False)$.
- Proof:
  - $\bigcup_i \tau^i(W) \subseteq \bigcup_i \tau^i(False)$:

$$
\begin{array}{rcl}
W & \subseteq & \bigcup_i \tau^i(False) \\
\tau(W) & \subseteq & \tau(\bigcup_i \tau^i(False)) = \bigcup_i \tau^i(False) \\
 & \vdots & \\
\tau^n(W) & \subseteq & \bigcup_i \tau^i(False) \\
 & \vdots & \\
\hline
\bigcup_i \tau^i(W) & \subseteq & \bigcup_i \tau^i(False)
\end{array}
$$

# Lemma 22 (cont.)

- $\bigcup_i \tau^i(False) \subseteq \bigcup_i \tau^i(W)$:

$$\begin{array}{rcl} False & \subseteq & W = \tau^0(W) \\ \tau(False) & \subseteq & \tau(W) \\ & \vdots & \\ \tau^n(False) & \subseteq & \tau^n(W) \\ & \vdots & \\ \hline \bigcup_i \tau^i(False) & \subseteq & \bigcup_i \tau^i(W) \end{array}$$

- So, to compute a least fixpoint, it is enough to start iterating with any approximation below the fixpoint.

# A Better Run Sketch

- Consider the calculation of $\mu Q_1.g_1(Q_1, \mu Q_2.g_2(Q_1, Q_2))$.
- We start with the initial approximation $Q_1^0 = \mathit{False}$.
- When computing $Q_2^{i\omega}$, we always begin with $Q_2^{i0} = Q_2^{(i-1)\omega}$.
  - Compute the inner fixpoint starting from $Q_2^{00} = \mathit{False}$ until we reach the fixpoint $Q_2^{0\omega}$.
  - $Q_1$ is increased to $Q_1^1 = g_1(Q_1^0, Q_2^{0\omega})$.
  - Compute the inner fixpoint starting from $Q_2^{10} = Q_2^{0\omega}$ until we reach the fixpoint $Q_2^{1\omega}$.
  - $Q_1$ is increased to $Q_1^2 = g_1(Q_1^1, Q_2^{1\omega})$.
  - ...
- This continues until we reach the fixpoint $Q_1^\omega$.

# A Better Run Sketch (cont.)

- Summary of the calculation of $\mu Q_1.g_1(Q_1, \mu Q_2.g_2(Q_1, Q_2))$:

| | | | |
|---|---|---|---|
| $Q_1^0$ $= \mathit{False}$ | $Q_2^{00}$ $= \mathit{False};$ | $Q_2^{01}$ $= g_2(Q_1^0, Q_2^{00});$ | $\ldots$ $Q_2^{0\omega}$ |
| $Q_1^1$ $= g_1(Q_1^0, Q_2^{0\omega})$ | $Q_2^{10}$ $= Q_2^{0\omega};$ | $Q_2^{11}$ $= g_2(Q_1^1, Q_2^{10});$ | $\ldots$ $Q_2^{1\omega}$ |
| $\vdots$ | $\vdots$ | | |
| $Q_1^{\omega-2}$ $= g_1(Q_1^{\omega-3}, Q_2^{(\omega-3)\omega})$ | $Q_2^{(\omega-2)0}$ $= Q_2^{(\omega-3)\omega};$ | $Q_2^{(\omega-2)1}$ $= g_2(Q_1^{(\omega-2)}, Q_2^{(\omega-2)0});$ | $\ldots$ $Q_2^{(\omega-2)\omega}$ |
| $Q_1^{\omega-1}$ $= g_1(Q_1^{\omega-2}, Q_2^{(\omega-2)\omega})$ | $Q_2^{(\omega-1)0}$ $= Q_2^{(\omega-2)\omega};$ | $Q_2^{(\omega-1)1}$ $= g_2(Q_1^{(\omega-1)}, Q_2^{(\omega-1)0});$ | $\ldots$ $Q_2^{(\omega-1)\omega}$ |
| $Q_1^{\omega}$ $= g_1(Q_1^{\omega-1}, Q_2^{(\omega-1)\omega})$ | | | |

- $Q_2^{0\omega} = g_2(Q_1^0, Q_2^{0\omega}) \subseteq g_2(Q_1^1, Q_2^{0\omega})$
- $Q_2^{0\omega} = \mu Q_2.g_2(Q_1^0, Q_2) \subseteq \mu Q_2.g_2(Q_1^1, Q_2) = Q_2^{1\omega}$

# Emerson-Lei Algorithm

1 **function** EL-Eval($f$, $e$)

2 **if** $f = p$ **then return** $\{s \mid p \in L(s)\}$;
3 **if** $f = Q$ **then return** $e(Q)$;
4 **if** $f = g_1 \wedge g_2$ **then**
5 **return** EL-Eval$(g_1, e) \cap$ EL-Eval$(g_2, e)$;
6 **if** $f = g_1 \vee g_2$ **then**
7 **return** EL-Eval$(g_1, e) \cup$ EL-Eval$(g_2, e)$;
8 **if** $f = \langle a \rangle g$ **then**
9 **return** $\{s \mid \exists t(s \xrightarrow{a} t$ and $t \in$ EL-Eval$(g, e))\}$;
10 **if** $f = [a]g$ **then**
11 **return** $\{s \mid \forall t(s \xrightarrow{a} t$ implies $t \in$ EL-Eval$(g, e))\}$;
12 **if** $f = \mu Q_i.g(Q_i)$ **then return** EL-Lfp$(g, e, Q_i)$;
13 **if** $f = \nu Q_i.g(Q_i)$ **then return** EL-Gfp$(g, e, Q_i)$;
14 **end function**

# Emerson-Lei Algorithm (cont.)

- The algorithm uses an array $A[1..N]$ to store the approximations to the fixpoints.
- Initially, $A[i]$ is set to *False* if the $i^{th}$ fixpoint formula is a least fixpoint and to *True* otherwise.
- The approximation values $A[i]$ are not reset when evaluating the subformula $\mu Q_i \,.\, g(Q_i)$ or $\nu Q_i \,.\, g(Q_i)$.

# Emerson-Lei Lfp

1 **function** EL-Lfp($g$, $e$, $Q_i$)

2 **forall** top-level greatest fixpoint subformulae $\nu Q_j.g'(Q_j)$ of $g$
3 **do** $A[j] \leftarrow True$;
4 **repeat**
5 $Q_{old} \leftarrow A[i]$;
6 $A[i] \leftarrow$ EL-Eval$(g, e[Q_i \leftarrow A[i]])$;
7 **until** $A[i] = Q_{old}$
8 **return** $A[i]$;
9 **end function**

# Emerson-Lei Gfp

1 **function** EL-Gfp($g$, $e$, $Q_i$)

2 **forall** top-level least fixpoint subformulae $\mu Q_j.g'(Q_j)$ of $g$
3 **do** $A[j] \leftarrow False$;
4 **repeat**
5 $Q_{old} \leftarrow A[i]$;
6 $A[i] \leftarrow$ EL-Eval$(g, e[Q_i \leftarrow A[i]])$;
7 **until** $A[i] = Q_{old}$
8 **return** $A[i]$;
9 **end function**

# Complexity Analysis

- In the naive algorithm, the innermost fixpoint requires $O(n^k)$ iterations, where $k$ is the maximum nesting depth of fixpoint operators.
- The number of iterations of Emerson-Lei algorithm is $O((|f| \cdot n)^d)$.
  - $|f|$ is an upper bound on the number of consecutive fixpoints of the same type in $f$.
  - The number of iterations for each such sequence is $O(|f| \cdot n)$, each fixpoint requiring at most $n$ iterations.
  - With $d$ alternating sequences, we have $O((|f| \cdot n)^d)$ iterations.

# Representing Formulae Using OBDDs

- The domain $S$ is encoded by the vector $\vec{x}$.
- Each atomic proposition $p$ has an OBDD associated with it, denoted $OBDD_p(\vec{x})$.
  - $\vec{y} \in \{0,1\}^n$ satisfies $OBDD_p$ iff $p \in L(\vec{y})$.
- Each transition $a$ has an OBDD associated with it, denoted $OBDD_a(\vec{x}, \vec{x}')$.
  - $(\vec{y}, \vec{z}) \in \{0,1\}^{2n}$ satisfies $OBDD_a$ iff $(\vec{y}, \vec{z}) \in a$.
- The environment is represented by a function *assoc*; $assoc[Q_i]$ gives the OBDD corresponding to the set of states associated with $Q_i$.
- $assoc\langle Q \leftarrow B_Q \rangle$ creates a new association by associating an OBDD $B_Q$ with $Q$.

# Representing Formulae Using OBDDs (cont.)

- The procedure $\mathtt{B}$ given below takes a $\mu$-calculus formula $f$ and an association list $assoc$ and returns an OBDD corresponding to the semantics of $f$.
  - $\mathtt{B}(p, assoc) = OBDD_p(\vec{x})$
  - $\mathtt{B}(Q_i, assoc) = assoc[Q_i]$
  - $\mathtt{B}(\neg f, assoc) = \neg \mathtt{B}(f, assoc)$
  - $\mathtt{B}(f \wedge g, assoc) = \mathtt{B}(f, assoc) \wedge \mathtt{B}(g, assoc)$
  - $\mathtt{B}(f \vee g, assoc) = \mathtt{B}(f, assoc) \vee \mathtt{B}(g, assoc)$
  - $\mathtt{B}(\langle a \rangle f, assoc) = \exists \vec{x}'(OBDD_a(\vec{x}, \vec{x}') \wedge \mathtt{B}(f, assoc)(\vec{x}'))$
  - $\mathtt{B}([a]f, assoc) = \mathtt{B}(\neg\langle a \rangle \neg f, assoc)$
  - $\mathtt{B}(\mu Q.f, assoc) = \mathtt{FIX}(f, assoc, OBDD_{False})$
  - $\mathtt{B}(\nu Q.f, assoc) = \mathtt{FIX}(f, assoc, OBDD_{True})$

# Representing Formulae Using OBDDs (cont.)

```
1  function FIX(f, assoc, B_Q)

2  bdd_result ← B_Q;
3  repeat
4      bdd_old ← bdd_result;
5      bdd_result ← B(f, assoc⟨Q ← bdd_old⟩);
6  until equal(bdd_old, bdd_result)
7  return bdd_result;
8  end function
```

# An example

- Let the state space $S$ be encoded by $n$ boolean variables $x_1, x_2, \ldots, x_n$.
- Let $OBDD_q(\vec{x})$ be the interpretation for $q$.
- The $OBDD$ corresponding to the transition $a$ is $OBDD_a(\vec{x}, \vec{x}')$.
- Given an association list *assoc* that pairs the $OBDD$ $B_Y(\vec{x})$ with $Y$.
- Consider the following formula:

$$f = \mu Z \ . \ ((q \wedge Y) \vee \langle a \rangle Z)$$

# An example (cont.)

- In the execution of `FIX`, $bdd_{result}$ is initially set to:

$$N^0(\vec{x}) = OBDD_{False}.$$

- At the end of the $i$-th iteration, the value of $bdd_{result}$ is given by:

$$N^{i+1}(\vec{x}) = (OBDD_q(\vec{x}) \wedge B_Y(\vec{x})) \vee \exists \vec{x}'(OBDD_a(\vec{x}, \vec{x}') \wedge N^i(\vec{x}')).$$

- The iteration stops when $N^i(\vec{x}) = N^{i+1}(\vec{x})$.

# Translating CTL into the $\mu$-Calculus

- Consider systems with just one transition $a$.
- The algorithm $\mathtt{Tr}$ takes as its input a CTL formula and outputs an equivalent $\mu$-calculus formula:
  - $\mathtt{Tr}(p) = p$
  - $\mathtt{Tr}(\neg f) = \neg \mathtt{Tr}(f)$
  - $\mathtt{Tr}(f \wedge g) = \mathtt{Tr}(f) \wedge \mathtt{Tr}(g)$
  - $\mathtt{Tr}(\mathbf{EX}\ f) = \langle a \rangle \mathtt{Tr}(f)$
  - $\mathtt{Tr}(\mathbf{E}[f\ \mathbf{U}\ g]) = \mu Y.(\mathtt{Tr}(g) \vee (\mathtt{Tr}(f) \wedge \langle a \rangle Y))$
  - $\mathtt{Tr}(\mathbf{EG}\ f) = \nu Y.(\mathtt{Tr}(f) \wedge \langle a \rangle Y)$

# Translating CTL into the $\mu$-Calculus (cont.)

- Example:

$$\begin{aligned} & \mathtt{Tr}(\mathbf{EG}\ \mathbf{E}[p\ \mathbf{U}\ q]) \\ = \ & \nu Y.(\mathtt{Tr}(\mathbf{E}[p\ \mathbf{U}\ q]) \wedge \langle a \rangle Y) \\ = \ & \nu Y.(\mu Z.(q \vee (p \wedge \langle a \rangle Z)) \wedge \langle a \rangle Y) \end{aligned}$$

- Any resulting $\mu$-calculus formula is closed.
- We can omit the environment $e$ from the translation.

# NP and co-NP

- We will see model checking $\mu$-calculus is in NP $\cap$ co-NP.
- A language $L$ is in NP if there exists a polynomial-time nondeterministic algorithm $M$ such that:
  - if $x \in L$, then $M(x) =$ "yes" for some computation path, and
  - if $x \notin L$, then $M(x) =$ "no" for all computation paths.
- A language $L$ is in co-NP if there exists a polynomial-time nondeterministic algorithm $M$ such that:
  - if $x \in L$, then $M(x) =$ "yes" for all computation paths, and
  - if $x \notin L$, then $M(x) =$ "no" for some computation path.
- co-NP $= \{L \mid \bar{L} \in$ NP$\}$.

# Relations between P, NP, and co-NP

- Current consensus (still open):
  - P $\neq$ NP
  - NP $\neq$ co-NP
  - P $\neq$ NP $\cap$ co-NP
- If an NP-complete problem is in co-NP, then NP $=$ co-NP.
  - Suppose $L$ is an NP-complete problem that is also in co-NP.
  - Let NTM $M$ decide $L$.
  - For any $L' \in$ NP, there is a reduction $R$ from $L'$ to $L$.
  - $L' \in$ co-NP as it is decided by NTM $M(R(\cdot))$.
  - Hence NP $\subseteq$ co-NP.
  - The other direction co-NP $\subseteq$ NP is symmetric.

# Complexity of Model Checking $\mu$-Calculus

- Problem: Given a finite model $M$, a state $s$, and a $\mu$-calculus formula $f$, does $M, s \models f$?
- Best known upper bound for this problem is NP $\cap$ co-NP.

# Model Checking $\mu$-Calculus Is in NP

- Consider the following nondeterministic algorithm:
  - Guess the greatest fixpoints and compute the least fixpoints by iteration.
  - The guess for a greatest fixpoint is checked to see that it really is a fixpoint.
  - Finally, check if the resulting set contains the given state.
- The greatest fixpoint must contain any verified guess.
- By monotonicity, this nondeterministic algorithm computes a subset of the real interpretation of the formula.
- There is a run of the algorithm which calculates the set of states satisfying the $\mu$-calculus formula.
- Consequently, the problem is in NP.

# Model Checking $\mu$-Calculus Is in co-NP

- Recall that co-NP $= \{L \mid \bar{L} \in$ NP$\}$.
- Consider the following nondeterministic algorithm:
  - Negate the input formula.
  - Apply the algorithm on the previous slide.
- Consequently, the problem is in co-NP.
- Hence, the problem is in NP $\cap$ co-NP.

# Open Problem

- Open Problem: Is there a polynomial model checking algorithm for the $\mu$-calculus?
- It is a long standing open problem.
- Clarke *et al.* conjecture NO in the book.
- If the problem was NP-complete, then NP = co-NP, which is believed to be unlikely.
- This suggests that it would be very difficult to prove the conjecture.