Propositional $\mu$-Calculus

Model: $M = (S, T, L)$, where
- $S$ - nonempty set of states;
- $T$ – a set of transitions, such that $\forall a \in T \cdot a \subseteq S \times S$
- $L : S \rightarrow S^{AP}$ gives the set of atomic propositions true in a state
- $VAR = \{ Q, Q_1, Q_2, \ldots \}$ – set of relational variables, where each $Q \in VAR$
can be assigned a subset of $S$

$\mu$-calculus formulae:
- If $p \in AP$, then $p$ is a formula.
- A relational variable is a formula.
- If $f$ and $g$ are formulas, then $\neg f, f \land g, f \lor g$ are formulas.
- If $f$ is a formula, and $a \in T$, then $[a]f$ and $< a > f$ are formulas.
- If $Q \in VAR$ and $f$ is a formula, then $\mu Q.f$ and $\nu Q.f$ are formulas, provided that $f$ is syntactically monotone in $Q$, i.e., all occurrences of $Q$ within $f$ fall under an even number of negations in $f$.

$\mu$-Calculus, Cont’d

- Variables: free or bound (by a fixpoint operator)
  E.g., $f(Q_1), \mu Q_1.f(Q_1)$
- $[a]f$ – “$f$ holds in all states reachable in one step by making an $a$-transition”
- $< a > f$ – “$f$ holds in at least one state reachable in one step by making an $a$ transition”
- $\mu$, $\nu$ – least and greatest fixpoints
- False – empty set of states
- True – all states $S$
- $s \xrightarrow{a} s'$ means $(s, s') \in a$
- $f$ – set of states where $f$ is true ($[[f]]_M e$, where $M$ - transition system, $e : VAR \rightarrow 2^S$ is an environment)
- $e[Q \leftarrow W]$ – new environment that is same as $e$ except that $e[Q \leftarrow W](Q) = W$
Semantics

- \([p]_{Me} = \{s \mid p \in L(s)\}\)
- \([Q]_{Me} = e(Q)\)
- \([-f]_{Me} = S - [f]_{Me}\)
- \([f \land g]_{Me} = [f]_{Me} \cap [g]_{Me}\)
- \([f \lor g]_{Me} = [f]_{Me} \cup [g]_{Me}\)
- \([< a > f]_{Me} = \{s \mid \exists t \cdot [s \xrightarrow{a} t \land t \in [f]_{Me}]\}\)
- \([[[a]f]_{Me} = \{s \mid \forall t \cdot [s \xrightarrow{a} t \Rightarrow t \in [f]_{Me}]\}\)
- \([\mu Q.f]_{Me}\) is the least fixpoint of the predicate transformer \(\tau : 2^S \rightarrow 2^S\) defined by \(\tau(W) = [f]_{Me}[Q \leftarrow W]\)
- \([\nu Q.f]_{Me}\) is the greatest fixpoint of the predicate transformer \(\tau : 2^S \rightarrow 2^S\) defined by \(\tau(W) = [f]_{Me}[Q \leftarrow W]\)

Relationship between \(\mu\)-calculus operators

\[
\begin{align*}
-[[a]f] & \equiv <a>\neg f \\
-[[<a>f] & \equiv [a]\neg f \\
-[[\mu Q.f](Q) & \equiv \nu Q.f(\neg Q) \\
-[[\nu Q.f](Q) & \equiv \mu Q.f(\neg Q)
\end{align*}
\]

How do we ensure existence of fixpoints?
Alternation Depth

Def: Alternation depth of a formula is the number of alternations between \( \mu \)-formulas and \( \nu \)-formulas along chains of nested fixpoint subformulas.

The definition is inductive:

- If \( \phi \) is not a fixpoint-formula then,

\[
ad(\phi) = \max\{ad(\psi) | \psi \text{ is a fixpoint-subformula of } \phi\}
\]

- else if \( \phi = \mu X.\psi \), then

\[
ad(\phi) = \max\{1, ad(\psi), 1 + \max\{ad(\chi) | \chi \text{ is open } \nu \text{-subformula of } \phi\}\}
\]

- else if \( \phi = \nu X.\psi \), then

\[
ad(\phi) = \max\{1, ad(\psi), 1 + \max\{ad(\chi) | \chi \text{ is open } \mu \text{-subformula of } \phi\}\}
\]

A \( \mu \)-calculus formula \( \phi \) is said to be alternation-free if \( ad(\phi) \leq 1 \).

Alternation-free \( \mu \)-calculus – a language of such \( \phi \)s.

Examples

\[
ad(\mu X. p \lor < a > X) = 1
\]
\[
ad(\nu X.((\nu Y. p \land [a]Y) \lor < a > X)) = 1
\]
\[
ad(\nu X. (p \land < a > \nu Y.(q \land [a]Y \lor < a > X))) = 1
\]
\[
ad(\nu X. \mu Y.((p \land X) \lor < a > Y)) = 2
\]

Note that the nesting depth (longest chain of fixpoint-subformulas of \( \phi \) that are nested in one another) of the first formula is 1, but for all the rest, it is 2.

Note: negating (and moving negation to atom. props) a \( \mu \)-calculus formula does not change its alternation depth.

Also note that fair CTL has alternation depth 2:

- Fair EG (with fairness condition \( h \))

\[
E_C G f = \nu Z.f \land EX(E[f U (f \land Z \land h)])
\]

\[
= \nu Z.(f \land < a > (\mu Y.(f \land Z \land h) \lor (f \land < a > Y)))
\]
Model-Checking Algorithm

1. function eval \((f, e)\)
2. if \(f = p\) then return \(\{s \mid p \in \mathcal{L}(s)\}\);
3. if \(f = g_1 \land g_2\) then
4. return \(\text{eval}(g_1, e) \cap \text{eval}(g_2, e)\);
5. if \(f = g_1 \lor g_2\) then
6. return \(\text{eval}(g_1, e) \cup \text{eval}(g_2, e)\);
7. if \(f = < a > g\) then
8. return \(\{s \mid \exists t \cdot [s \xrightarrow{a} t \text{ and } t \in \text{eval}(g, e)]\}\);
9. if \(f = [a]g\) then
10. return \(\{s \mid \forall t \cdot [s \xrightarrow{a} t \text{ implies } t \in \text{eval}(g, e)]\}\);

Model-Checking Algorithm (Cont’d)

11. if \(f = \mu Q.g(Q)\) then
12. \(Q_{\text{val}} := \text{False}\);
13. repeat
14. \(Q_{\text{old}} := Q_{\text{val}}\);
15. \(Q_{\text{val}} := \text{eval}(g, e[Q \leftarrow Q_{\text{val}}])\);
16. until \(Q_{\text{val}} = Q_{\text{old}}\);
17. return \(Q_{\text{val}}\);
18. if \(f = v Q.g(Q)\) then
19. \(Q_{\text{val}} := \text{True}\);
20. repeat
21. \(Q_{\text{old}} := Q_{\text{val}}\);
22. \(Q_{\text{val}} := \text{eval}(g, e[Q \leftarrow Q_{\text{val}}])\);
23. until \(Q_{\text{val}} = Q_{\text{old}}\);
24. return \(Q_{\text{val}}\);
25. end function
**Complexity**

1. Each loop executes at most $n + 1$ times ($n = |S|$)
2. Each iteration does a recursive call to evaluate the body of fixpoint with a different value for the fixpoint variable
3. It can also lead to recursive calls...

Complexity: $O(n^k)$ iterations of the fixpoint, where $k$ – maximum nesting depth of fixpoint operators in the formula.

Each iteration: $O(|M| \times |f|)$, where

$|M| = |S| + \sum_{a \in T} |a|$

Overall complexity: $O(|M| \times |f| \times n^k)$

---

**A Better Algorithm [Emerson, Lai]**

Goal: decrease the number of fixpoint iterations to $O(|f| \times n^d)$, where $d$ – alternation depth of $f$.

Idea: exploit sequences of fixpoints that have the same type to reduce the complexity of the algorithm:

- It is unnecessary to reinitialize 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. Similar, for greatest fixpoint.
Emerson-Lai Algorithm

11. if $f = \mu Q_i . g(Q_i)$ then
12. for all top-level greatest fixpoint subformulas $\nu Q_j . g(Q_j)$ of $g$
14. repeat
15. $Q_{old} := A[i]$;
16. $A[i] := \text{eval}(g, e[Q_i \leftarrow A[i]])$;
17. until $A[i] = Q_{old}$;
18. return $A[i]$;

Emerson-Lai Cont’d

19. if $f = \nu Q_i . g(Q_i)$ then
20. for all top-level least fixpoint subformulas $\mu Q_j . g(Q_j)$ of $g$
22. repeat
23. $Q_{old} := A[i]$;
25. until $A[i] = Q_{old}$;
26. return $A[i]$;
27. end function
Complexity

1. $|f|$ – upper bound on the number of consecutive fixpoints of the same type in $f$
2. Number of iterations for each such sequences is $O(|f| \times n)$ instead of $n|f|$ as before
3. Computation is reinitialized at the boundary between two sequences of different types

Overall number of iterations: $O((|f| \times n)^d)$

Moreover, complexity of model-checking $\mu$-calculus is in NP $\cap$ co-NP (see book)

[Sterling'03] Complexity of model-checking $\mu$-calculus is in P!

$\mu$-calculus and CTL

Translation of CTL into $\mu$-calculus ($a$ is the only transition):

\[
\begin{align*}
Tr(p) &= p \\
Tr(\neg f) &= \neg Tr(f) \\
Tr(f \land g) &= Tr(f) \land Tr(g) \\
Tr(EX f) &= <a> Tr(f) \\
Tr(E[fUg]) &= \mu Y.(Tr(g) \lor (Tr(f) \land <a> Y)) \\
Tr(EG f) &= \nu Y.(Tr(f) \land <a> Y)
\end{align*}
\]

Any resulting $\mu$-calculus formula is closed; so, omit environment $e$ from translation.
Example: \( Tr(EGE[pUq]) = \nu Y. (\mu Z. (q \lor (p \land <a>Z)) \land <a>Y) \)

Theorem: Let \( M = (S, T, L) \) be a Kripke structure. Assume that the transition \( a \) in the translation algorithm \( Tr \) is the relation \( T \) of the Kripke structure. Let \( f \) be a CTL formula. Then, for all \( s \in S \),

\[
M, s \models f \iff s \in [[Tr(f)]]_M
\]