Propositional μ-Calculus

Model: $M = (S, T, L)$, where

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

μ-calculus formulae:

- If $p \in AP$, then p is a formula.
- A relational variable is a formula.
- If f and g are formulas, then $\neg f$, $f \wedge g$, $f \vee g$ 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

μ-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"
- μ, ν – 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, $e : VAR \to 2^S$ is an *environment*)
- $e[Q \leftarrow W]$ – new environment that is same as e except that $e[Q \leftarrow W](Q) = W$

Semantics

- $[[p]]_M e = \{s \mid p \in L(s)\}$
- $[[Q]]_M e = e(Q)$
- $[[\neg f]]_M e = S - [[f]]_M e$
- $[[f \wedge g]]_M e = [[f]]_M e \cap [[g]]_M e$
- $[[f \vee g]]_M e = [[f]]_M e \cup [[g]]_M e$
- $[[< a > f]]_M e = \{s \mid \exists t \cdot [s \xrightarrow{a} t \ \wedge \ t \in [[f]]_M e]\}$
- $[[[a] f]]_M e = \{s \mid \forall t \cdot [s \xrightarrow{a} t \Rightarrow t \in [[f]]_M e]\}$
- $[[\mu Q. f]]_M e$ is the least fixpoint of the predicate transformer $\tau : 2^S \rightarrow 2^S$

defined by $\tau(W) = [[f]]_M e[Q \leftarrow W]$

- $[[\nu Q. f]]_M e$ is the greatest fixpoint of the predicate transformer $\tau : 2^S \rightarrow 2^S$

defined by $\tau(W) = [[f]]_M e[Q \leftarrow W]$

Relationship between μ-calculus operators

$$\begin{array}{lcl} \neg [a] f & \equiv & < a > \neg f \\ \neg < a > f & \equiv & [a] \neg f \\ \neg \mu Q. f(Q) & \equiv & \nu Q. \neg f(\neg Q) \\ \neg \nu Q. f(Q) & \equiv & \mu Q. \neg f(\neg Q) \end{array}$$

How do we ensure existence of fixpoints?

Alternation Depth

Def: *Alternation depth* of a formula is the number of alternations between μ-formulas and ν-formulas along chains of nested fixpoint subformulas.

The definition is inductive:

- If φ is not a fixpoint-formula then,

$$ad(\varphi) = max\{ad(\psi) | \psi \text{ is a fixpoint-subformula of } \varphi\}$$

- else if $\varphi = \mu X.\psi$, then

$$ad(\varphi) = max\{1, ad(\psi), 1 + max\{ad(\chi) \mid \chi \text{ is open } \nu\text{-subformula of } \varphi\}\}$$

- else if $\varphi = \nu X.\psi$, then

$$ad(\varphi) = max\{1, ad(\psi), 1 + max\{ad(\chi) \mid \chi \text{ is open } \mu\text{-subformula of } \varphi\}\}$$

A μ-calculus formula φ is said to be *alternation-free* if $ad(\varphi) \leq 1$.
Alternation-free μ-calculus – a language of such φs.

Examples

$$\begin{array}{lcl}
ad(\mu X.p \vee < a > X) & = & 1 \\
ad(\nu X.((\nu Y.p \wedge [a]Y) \vee < a > X)) & = & 1 \\
ad(\nu X.(p \wedge < a > \nu Y.(q \wedge [a]Y \vee < a > X)) & = & 1 \\
ad(\nu X.\mu Y.((p \wedge X) \vee < a > Y)) & = & 2
\end{array}$$

Note that the *nesting depth* (longest chain of fixpoint-subformulas of φ 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 μ-calculus formula does not change its alternation depth.

Also note that *fair CTL* has alternation depth 2:

- Fair EG (with fairness condition h)

$$\begin{array}{rl}
& E_C G f = \nu Z.f \wedge EX(E[f\, U\, (f \wedge Z \wedge h)]) \\
= & \nu Z.(f \wedge < a > (\mu Y.(f \wedge Z \wedge h) \vee (f \wedge < a > Y)))
\end{array}$$

Model-Checking Algorithm

function eval (f, e)
if $f = p$ then return $\{s \mid p \in L(s)\}$;
if $f = g_1 \wedge g_2$ then
  return eval(g_1, e) $\cap$ eval(g_2, e);
if $f = g_1 \vee g_2$ then
  return eval(g_1, e) $\cup$ eval(g_2, e);
if $f = < a > g$ then
  return $\{s \mid \exists t \cdot [s \xrightarrow{a} t$ and $t \in$ eval$(g, e)]\}$;
if $f = [a]g$ then
  return $\{s \mid \forall t \cdot [s \xrightarrow{a} t$ implies $t \in$ eval$(g, e)]\}$;

Model-Checking Algorithm (Cont'd)

if $f = \mu Q.g(Q)$ then
  Q_{val} := *False*;
  repeat
    Q_{old} := Q_{val};
    Q_{val} := eval($g, e[Q \leftarrow Q_{val}]$);
  until $Q_{val} = Q_{old}$;
  return Q_{val};
if $f = \nu Q.g(Q)$ then
  Q_{val} := *True*;
  repeat
    Q_{old} := Q_{val};
    Q_{val} := eval($g, e[Q \leftarrow Q_{val}]$);
  until $Q_{val} = Q_{old}$;
  return Q_{val};
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
$|M| = |S| + \Sigma_{a \in T} |a|$

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

if $f = \mu Q_i.g(Q_i)$ then
  for all top-level greatest fixpoint subformulas $\nu Q_j.g'(Q_j)$ of g
    do A[j] := *True*;
  repeat
    Q_{old} := A[i];
    A[i] := eval($g, e[Q_i \leftarrow A[\mathrm{i}]]$);
  until A[i] = Q_{old};
  return A[i];

Emerson-Lai Cont'd

if $f = \nu Q_i.g(Q_i)$ then
  for all top-level least fixpoint subformulas $\mu Q_j.g'(Q_j)$ of g
    do A[j] := *False*;
  repeat
    Q_{old} := A[i];
    A[i] := eval($g, e[Q_i \leftarrow A[\mathrm{i}]]$);
  until A[i] = Q_{old};
  return A[i];
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 μ-calculus is in NP $\cap$ co-NP (see book)

[Sterling'03] Complexity of model-checking μ-calculus is in P!

μ-calculus and CTL

Translation of CTL into μ-calculus (a is the only transition):

$$
\begin{array}{lcl}
Tr(p) & = & p \\
Tr(\neg f) & = & \neg Tr(f) \\
Tr(f \wedge g) & = & Tr(f) \wedge Tr(g) \\
Tr(EXf) & = & < a > Tr(f) \\
Tr(E[fUg]) & = & \mu Y.(Tr(g) \vee (Tr(f) \wedge < a > Y)) \\
Tr(EGf) & = & \nu Y.(Tr(f) \wedge < a > Y)
\end{array}
$$

Any resulting μ-calculus formula is closed; so, omit environment e from translation.

μ-calculus and CTL, Cont'd

Example: $Tr(EGE[pUq])) =$
$\nu Y.(\mu Z.(q \vee (p \wedge \ <a> Z)) \wedge \ <a> Y)$

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 $s \in S$,

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

food for slide eater