Algorithms for Model Checking (2IW55)

Lecture 6
The $\mu$-Calculus
Chapter 7

Tim Willemse
(timw@win.tue.nl)
http://www.win.tue.nl/~timw
HG 6.81
Outline

$\mu$-Calculus: syntax and semantics

Examples

Complexity

Emerson-Lei Algorithm

Embedding CTL-formulae

Conclusions

Exercise
Recall: symbolic model checking for CTL was based on fixed points.

Idea of $\mu$-calculus: add fixed point operators as primitives to basic modal logic.

- $\mu$-calculus is very expressive (subsumes CTL, LTL, CTL*).
- $\mu$-calculus is very pure ("assembly language" for modal logic, cf: $\lambda$-calculus for functional programming).
- drawback: lack of intuition.
- fragments of the $\mu$-calculus are the basis for practical model checkers, such as $\mu$CRL, mCRL2, CADP, Concurrency Workbench.
Kripke Structures and Labelled Transition Systems

Mix of Kripke Systems and Labelled Transition Systems: $M = \langle S, \text{Act}, R, L \rangle$ over a set $AP$ of atomic propositions:

- $S$ is a set of states
- $\text{Act}$ is a set of action labels
- $R$ is a labelled transition relation: $R \subseteq S \times \text{Act} \times S$
- $L$ is a labelling: $L \in S \rightarrow 2^{AP}$

Notation: $s \xrightarrow{a} t$ denotes $(s, a, t) \in R$

Special cases:

- Kripke Structures: $\text{Act}$ is a singleton (only one transition relation)
- LTS (process algebra): $AP$ is empty (only propositions true and false)

In the book, the set of labels $\text{Act}$ is not made explicit, but $R \subseteq 2^{S \times S}$ is a set of transitions: for each $a \in R$, $a \subseteq S \times S$. 
Let the following sets be given: \( AP \) (atomic propositions), \( Act \) (action labels) and \( Var \) (formal variables).

The syntax of \( \mu \)-calculus formulae \( f \) is defined by the following grammar:

\[
f ::= p \mid X \mid \neg f \mid f \land f \mid f \lor f \mid \exists a f \mid \forall a f \mid \mu X. f \mid \nu X. f
\]

Note:
- \( p \in AP, X \in Var, a \in Act \).
- \( \exists a f \) means “for all direct \( a \)-successors, \( f \) holds”.
- \( \forall a f \) means “for some direct \( a \)-successor, \( f \) holds”.
- We only consider fixed point formulae \( \nu X. f \) if \( X \) occurs under an even number of negations (\( \neg \)) in \( f \).
Some notation and terminology:

- "X occurs in f only under an even number of ¬-symbols" is called the syntactic monotonicity criterion. This criterion ensures the (semantic) existence of fixed points.

- An occurrence of X is **bound** by a surrounding fixed point symbol \( \mu \) X \((\mu \in \{\mu, \nu\})\). Unbound occurrences of X are called **free**.

- A **closed** formula has no free variables. If it has free variables, a formula is called **open**.

- An **environment** e interprets the free formal variables X as a set of states:
  - Mixed Kripke Structure \( M = \langle S, Act, R, L \rangle \)
  - \( e : Var \rightarrow 2^S \)
  - \( e[X := V] \) is a new environment like e, but X is set to V:

\[
e[X := V](Y) := \begin{cases} 
V & \text{if } Y = X \\
e(Y) & \text{otherwise}
\end{cases}
\]
Fix a system: $M = \langle S, Act, R, L \rangle$

- The semantics of open formulae is only defined if we know the values of the free variables.
- The semantics of a $\mu$-Calculus formula $f$ in the context of environment $e$ is the set of states where $f$ holds:

$$
\begin{align*}
[-f]_e &= S \setminus [f]_e \\
[p]_e &= \{ s \mid p \in L(s) \} \\
[f \land g]_e &= [f]_e \cap [g]_e \\
[f \lor g]_e &= [f]_e \cup [g]_e \\
[\langle a \rangle f]_e &= \{ s \mid \forall t. s \xrightarrow{a} t \Rightarrow t \in [f]_e \} \\
[\nu X. f]_e &= gfp(Z \mapsto [f]_e[X:=Z]) \\
[\mu X. f]_e &= lfp(Z \mapsto [f]_e[X:=Z])
\end{align*}
$$

The semantics immediately gives rise to a **naive algorithm** for model checking $\mu$-calculus (compute $lfp$ and $gfp$ by iteration).
Outline

- $\mu$-Calculus: syntax and semantics
- Examples
- Complexity
- Emerson-Lei Algorithm
- Embedding CTL-formulae
- Conclusions
- Exercise
Examples

- Not a $\mu$-calculus formula: $\mu X. \neg X$
- Let $Act = \{a\}$:
  - $E G f$ ................................................................. $v X. f \land (a) X$
  - $E [f U g]$ ............................................................... $\mu X. g \lor (f \land (a) X)$
  - Every $p$ is inevitably followed by a $q$: $v X_1. \left( (p \Rightarrow (\mu X_2. q \lor [a] X_2)) \land [a] X_1 \right)$
- Special case: $X_1$ does not occur within the scope of $\mu X_2$.
- The last formula can therefore be evaluated “inside-out”:

\[
\begin{align*}
X_2^0 & = \text{false} \\
X_2^1 & = q \lor [a] X_2^0 \\
X_2^2 & = q \lor [a] X_2^1 \\
\ldots & \\
X_2^\omega & = q \lor [a] X_2^\omega
\end{align*}
\]

\[
\begin{align*}
X_1^0 & = \text{true} \\
X_1^1 & = (p \Rightarrow X_2^\omega) \land [a] X_1^0 \\
X_1^2 & = (p \Rightarrow X_2^\omega) \land [a] X_1^1 \\
\ldots & \\
X_1^\omega & = (p \Rightarrow X_2^\omega) \land [a] X_1^\omega
\end{align*}
\]
Examples

A more difficult case

- On some path, $h$ holds infinitely often: $\nu X_1. \langle a \rangle (\mu X_2. (X_1 \land h) \lor \langle a \rangle X_2)$
- Problem: the inner fixed point depends crucially on $X_1$.

\[
\begin{align*}
X_1^0 &= \text{true} \\
X_1^0 &= \langle a \rangle X_2^{0\omega} \\
X_1^0 &= \text{false} \\
X_2^{00} &= (X_1^0 \land h) \lor \langle a \rangle X_2^{00} \\
X_2^{01} &= (X_1^0 \land h) \lor \langle a \rangle X_2^{01} \\
X_2^{02} &= (X_1^0 \land h) \lor \langle a \rangle X_2^{02} \\
X_2^{0\omega} &= (X_1^0 \land h) \lor \langle a \rangle X_2^{0\omega} \\
X_2^{10} &= (X_1^1 \land h) \lor \langle a \rangle X_2^{10} \\
X_2^{1\omega} &= (X_1^1 \land h) \lor \langle a \rangle X_2^{1\omega} \\
\ldots
\end{align*}
\]
Outline

μ-Calculus: syntax and semantics

Examples

Complexity

Emerson-Lei Algorithm

Embedding CTL-formulae

Conclusions

Exercise
Complexity of naive $\mu$-Calculus algorithm

- We check formula $f$ with at most $k$ nested fixed points on the Kripke Structure $M = \langle S, R, Act, L \rangle$.
- In the previous example:
  - The outermost (greatest) fixed point can decrease at most $|S|$ times (recall that $S$ is finite)
  - Each time, the innermost fixed point of formula $f$ is evaluated at most $|S|^k$ times, where $k$ is the maximum number of nested fixed points in $f$.
- In general: the innermost fixed point of formula $f$ is evaluated at most $|S|^k$ times, where $k$ is the maximum number of nested fixed points in $f$.
- Each iteration requires up to $|M| \times |f|$ steps.
- **Total time complexity** of naive algorithm: $O((|S| + |R|) \times |f| \times |S|^k)$.

A more careful analysis will yield a more optimal treatment for nested fixed points of the same type.
A μ-calculus formula is in **positive normal form** if negations occur only before propositions.

To transform a formula into positive normal form, negations can be pushed inside using logical *dualities*:

\[
\begin{align*}
\neg\neg f & \iff f \\
\neg(f \lor g) & \iff (\neg f) \land (\neg g) \\
\neg(f \land g) & \iff (\neg f) \lor (\neg g) \\
\neg([a]f) & \iff \langle a \rangle (\neg f) \\
\neg(\langle a \rangle f) & \iff [a] (\neg f) \\
\neg(\mu X.f(X)) & \iff \nu X.\neg f(\neg X) \\
\neg(\nu X.f(X)) & \iff \mu X.\neg f(\neg X)
\end{align*}
\]

Due to syntactic monotonicity, single negations in front of formal variables cannot arise.

Hence, the result is a positive normal form.

**Check:** the result is logically equivalent.
The complexity of a $\mu$-calculus formula depends on the fixed points (cf. the complexity of first-order formulae depends on the quantifiers)

- **Nesting Depth:**
  maximum number of nested fixed points in a positive normal form

\[
\begin{align*}
ND(f) & := 0 & \text{for } f \in \{p, \neg p, X\} \\
ND(\circ f) & := ND(f) & \text{for } \circ \in \{[a], \langle a \rangle\} \\
ND(f \square g) & := \max(ND(f), ND(g)) & \text{for } \square \in \{\land, \lor\} \\
ND(\mu_{i} X.f) & := 1 + ND(f) & \text{for } \mu_{i} \in \{\mu, \nu\}
\end{align*}
\]

- Example: \( ND\left((\mu X_{1}. \nu X_{2}. X_{1} \lor X_{2}) \land (\mu X_{3}. \mu X_{4}. (X_{3} \land \mu X_{5}. p \lor X_{5}))\right) \)
The complexity of a $\mu$-calculus formula depends on the fixed points (cf. the complexity of first-order formulae depends on the quantifiers)

- **Nesting Depth:**
  maximum number of nested fixed points in a positive normal form

\[
ND(f) := 0 \quad \text{for } f \in \{p, \neg p, X\}
\]
\[
ND(\circ f) := ND(f) \quad \text{for } \circ \in \{[a], \langle a \rangle\}
\]
\[
ND(f \Box g) := \max(ND(f), ND(g)) \quad \text{for } \Box \in \{\land, \lor\}
\]
\[
ND(\nu \mu X.f) := 1 + ND(f) \quad \text{for } \nu, \mu \in \{\mu, \nu\}
\]

- **Example:** $ND\left((\mu X_1. \nu X_2. X_1 \lor X_2) \land (\mu X_3. \mu X_4. (X_3 \land \mu X_5. p \lor X_5))\right) = 3$
Complexity

- **Alternation Depth**: number of alternating fixed points of a formula in positive normal form.

\[
\begin{align*}
AD(f) & := 0 & \text{for } f \in \{p, \neg p, X\} \\
AD(\oplus f) & := AD(f) & \text{for } \oplus \in \{[a], \langle a \rangle\} \\
AD(f \Box g) & := \max(AD(f), AD(g)) & \text{for } \Box \in \{\land, \lor\} \\
AD(\mu X.f) & := 1 + \max\{AD(g) \mid g \text{ is a } \nu\text{-subformula of } f\} \\
AD(\nu X.f) & := 1 + \max\{AD(g) \mid g \text{ is a } \mu\text{-subformula of } f\}
\end{align*}
\]

- **Examples**:

\[
\begin{align*}
AD\left(\mu X_1. \nu X_2. X_1 \lor X_2 \land (\mu X_3.\mu X_4. (X_3 \land \mu X_5.p \lor X_5))\right)
\end{align*}
\]

\[
\begin{align*}
AD\left(\mu X_1. \nu X_2. X_1 \lor X_2 \land (\mu X_3.\nu X_4. (X_3 \land \mu X_5.p \lor X_5))\right)
\end{align*}
\]
Complexity

- **Alternation Depth**: number of alternating fixed points of a formula in positive normal form.

\[
\begin{align*}
AD(f) & := 0 & \text{for } f \in \{p, \neg p, X\} \\
AD(\@ f) & := AD(f) & \text{for } \@ \in \{[a], \langle a \rangle\} \\
AD(f \Box g) & := \max(AD(f), AD(g)) & \text{for } \Box \in \{\land, \lor\} \\
AD(\mu X.f) & := 1 + \max\{AD(g) \mid g \text{ is a } \nu\text{-subformula of } f\} \\
AD(\nu X.f) & := 1 + \max\{AD(g) \mid g \text{ is a } \mu\text{-subformula of } f\}
\end{align*}
\]

- **Examples**:

\[
AD\left((\mu X_1. \nu X_2. X_1 \lor X_2) \land (\mu X_3.\mu X_4. (X_3 \land \mu X_5.p \lor X_5))\right) = 2
\]

\[
AD\left((\mu X_1. \nu X_2. X_1 \lor X_2) \land (\mu X_3.\nu X_4. (X_3 \land \mu X_5.p \lor X_5))\right) = 3
\]
Complexity

- **Dependent Alternation Depth (dAD):** number of alternating fixed points, such that the innermost fixed point depends on the outermost.

- The definition of \( dAD \) is identical to \( AD \), except for

\[
dAD(\mu X.f) := \max(dAD(f), 1 + \max\{dAD(g) \mid g \text{ is a } \nu\text{-subformula of } f \text{ and } X \text{ occurs in } g\})
\]

\[
dAD(\nu X.f) := \max(dAD(f), 1 + \max\{AD(g) \mid g \text{ is a } \mu\text{-subformula of } f \text{ and } X \text{ occurs in } g\})
\]

- **Examples:**

\[
dAD\left((\mu X_1. \nu X_2. X_1 \lor X_2) \land (\mu X_3. \mu X_4. (X_3 \land \mu X_5. p \lor X_5))\right)
\]

\[
dAD\left((\mu X_1. \nu X_2. X_1 \lor X_2) \land (\mu X_3. \nu X_4. (X_3 \land \mu X_5. p \lor X_5))\right)
\]
Complexity

- **Dependent Alternation Depth (dAD):** number of alternating fixed points, such that the innermost fixed point depends on the outermost.

- The definition of $dAD$ is identical to $AD$, except for

\[
\begin{align*}
dAD(\mu X. f) & := \max(dAD(f), 1 + \max\{dAD(g) \mid g \text{ is a } \nu\text{-subformula of } f \text{ and } X \text{ occurs in } g\}) \\
dAD(\nu X. f) & := \max(dAD(f), 1 + \max\{AD(g) \mid g \text{ is a } \mu\text{-subformula of } f \text{ and } X \text{ occurs in } g\})
\end{align*}
\]

- **Examples:**

\[
\begin{align*}
dAD\left((\mu X_1. \nu X_2. X_1 \lor X_2) \land (\mu X_3. \mu X_4. (X_3 \land \mu X_5.p \lor X_5))\right) &= 2 \\
dAD\left((\mu X_1. \nu X_2. X_1 \lor X_2) \land (\mu X_3. \nu X_4. (X_3 \land \mu X_5.p \lor X_5))\right) &= 2
\end{align*}
\]
Outline

\( \mu \)-Calculus: syntax and semantics

Examples

Complexity

Emerson-Lei Algorithm

Embedding CTL-formulae

Conclusions

Exercise
Emerson-Lei Algorithm

- Given a **finite** set $S$ and a **monotonic** $\tau : 2^S \rightarrow 2^S$ in the partial order $(2^S, \subseteq)$.
- We used to compute the least fixed point from $\emptyset$:

$$\emptyset \subseteq \tau(\emptyset) \subseteq \tau^2(\emptyset) \subseteq \ldots \subseteq \tau^i(\emptyset) = \tau^{i+1}(\emptyset)$$

then $\mu X.\tau(X) = \tau^i(\emptyset)$

- Actually, instead of $\emptyset$, we can start in any set known to be **smaller** than the fixed point:
  - Assume $W \subseteq \mu X.\tau(X)$, so we have:

$$\emptyset \subseteq W \subseteq \tau^i(\emptyset)$$

  - By monotonicity and the definition of fixed points:

$$\tau^i(\emptyset) \subseteq \tau^i(W) \subseteq \tau^{2i}(\emptyset) = \tau^i(\emptyset)$$

  - So if $W \subseteq \mu X.\tau(X)$ we compute the least fixed point as:

$$W, \tau(W), \tau^2(W), \ldots, \tau^j(W) = \tau^{j+1}(W)$$

This converges at some $j \leq i$ (may be $j < i$)
The observations on the previous slide can speed up computations of nested fixed points.

Consider two nested  \( \mu \)-fixed points: 
\[
\mu X_1.f(X_1, \mu X_2.g(X_1, X_2))
\]

Start approximation of \( X_1 \) and \( X_2 \) with \( X_1^0 = X_2^0 = \text{false} \):

\[
\begin{align*}
X_1^0 &= \text{false} \\
X_2^0 &= \text{false} \\
X_2^{01} &= g(X_1^0, X_2^{00}) \\
\vdots \\
X_2^{0\omega} &= g(X_1^0, X_2^{0\omega}) \\
X_1^1 &= f(X_1^0, X_2^{0\omega})
\end{align*}
\]

Clearly, \( X_1^0 \subseteq X_1^1 \), so also \( X_2^{0\omega} = \mu X_2.g(X_1^0, X_2) \subseteq \mu X_2.g(X_1^1, X_2) = X_2^{1\omega} \).

So, we approximating \( X_2 \) can start at \( X_2^{0\omega} \) instead of at \( \text{false} \):

\[
\begin{align*}
X_2^{10} &= X_2^{0\omega} \\
\vdots \\
X_2^{1\omega} &= g(X_1^1, X_2^{1\omega}) \\
X_1^2 &= f(X_1^1, X_2^{1\omega})
\end{align*}
\]
Emerson-Lei Algorithm

Given:
- Mixed Kripke Structure: $M = \langle S, R, Act, L \rangle$
- A $\mu$-Calculus formula $f$ and an environment $e$

Returns: $[f]_e$, the set of states in $S$ where $f$ holds.

Idea:
- The function $\text{EVAL}(f)$ proceeds by recursion on $f$, using iteration for the fixed points.
- The value of the current approximation for variable $X_i$ is stored in array $A[i]$, in order to reuse it in later iterations.
- Reset $A[i]$ only if:
  - a higher $X_j$ of different sign changed, and
  - $\nu X_i. f$ contains free variables.
Emerson-Lei algorithm

Initialisation:

for all variables $X_i$ do
    if $X_i$ is bound by a $\mu$ then $A[i] := \text{false}$;
    else if $X_i$ is bound by a $\nu$ then $A[i] := \text{true}$;
    else $A[i] := e(X_i)$
end if
end for
function EVAL(f)
    if $f = X_i$ then return $A[i]$
    else if $f = g_1 \lor g_2$ then return $\text{EVAL}(g_1) \cup \text{EVAL}(g_2)$
    else if . . . then . . .
    else if $f = \mu X_i.g(X_i)$ then
        if the surrounding binder of $f$ is a $\nu$ then
            for all open subformulae of $f$ of the form $\mu X_k.g$ do $A[k] := false$
        end for
    end if
    repeat
        $X_{old} := A[i]$;
        $A[i] := \text{EVAL}(g)$;
        until $A[i] = X_{old}$
    return $A[i]$
end if
end function
Emerson-Lei algorithm

Given a formula $\nu X_1.\nu X_2.\mu X_3.\mu X_4.(X_1 \lor X_2 \lor (\mu X_5.X_5 \land p))$

- When computing $\nu X_2$, $\mu X_4$ and $\mu X_5$: no reset is needed because the surrounding binder has the same sign.
- When computing $X_3$:
  - Reset $X_3$, $X_4$: their subformula contains $X_1$ and $X_2$ as free variables
  - Do not reset $X_5$: the subformula $(\mu X_5.X_5 \land p)$ is closed

Modifications with respect to the book (p. 105):

- We identified $e$ and $A[i]$ (they play the same role)
- The restriction to reset open formulae only makes the algorithm more efficient. This is essential for CTL (see later).
- The book is wrong: the reset of $A[j]$ should occur within the repeat-until loop. It resets the wrong fixed points. We went back to the original Emerson and Lei algorithm (1986).
Emerson-Lei algorithm

Complexity analysis

- Let formula $f$ be given, with dependent alternation depth $d_{AD}(f) = d$.
- Let the Kripke Structure be $\langle S, Act, R, L \rangle$.
- Take a block of fixed points of the same type:
  - its length is at most $|f|$.
  - the value of each fixed point in it can grow/shrink at most $|S|$ times.
- In total, the innermost block will have no more than $(|f| \cdot |S|)^d$ iterations of the repeat-loop.
- Each iteration requires time at most $\mathcal{O}(|f| \cdot (|S| + |R|))$.
- Hence: the overall complexity of the Emerson-Lei algorithm is $\mathcal{O}(|f| \cdot (|S| + |R|) \cdot (|f| \cdot |S|)^d)$.
Outline

$\mu$-Calculus: syntax and semantics

Examples

Complexity

Emerson-Lei Algorithm

Embedding CTL-formulae

Conclusions

Exercise
Again, assume $Act = \{a\}$. Given the fixed point characterisation of CTL, there is a straightforward translation of CTL to the $\mu$-calculus:

- $Tr(p) = p$
- $Tr(\neg f) = \neg Tr(f)$
- $Tr(f \land g) = Tr(f) \land Tr(g)$
- $Tr(\exists X f) = \langle a \rangle Tr(f)$
- $Tr(\exists G f) = \nu Y. (Tr(f) \land \langle a \rangle Y)$
- $Tr(\exists [f \cup g]) = \mu Y. (Tr(g) \lor (Tr(f) \land \langle a \rangle Y))$

Note:

- $Tr(f)$ is syntactically monotone
- $Tr(f)$ is a closed $\mu$-calculus formula
- $dAD(Tr(f)) \leq 1$, which is called the alternation free fragment of the $\mu$-calculus
- $AD(Tr(f))$ is not bounded!
Outline

μ-Calculus: syntax and semantics

Examples

Complexity

Emerson-Lei Algorithm

Embedding CTL-formulae

Conclusions

Exercise
Conclusions

- the $\mu$-calculus incorporates least and greatest fixed points directly in the logic.
- the naive algorithm is exponential in the nesting depth of fixed points.
- a careful analysis leads to an algorithm which is exponential in the (dependent) alternation depth only,
- Hence: alternation free $\mu$-calculus is linear in the Kripke Structure and polynomial in the formula.
- CTL translates into the alternation free fragment of the $\mu$-calculus.
- for the latter we essentially needed the dependent alternation depth.
- fairness constraints typically lead to one extra alternation ($dAD(f) = 2$)
Outline

$\mu$-Calculus: syntax and semantics

Examples

Complexity

Emerson-Lei Algorithm

Embedding CTL-formulae

Conclusions

Exercise
Consider the following $\mu$-calculus formula $\phi$ and LTS $\mathcal{L}$:

$$\phi := \nu X. ([a] X \land \nu Y. \mu Z. (\langle b \rangle Y \lor \langle a \rangle Z))$$

- Compute the set of states where $\phi$ holds with the naive algorithm (give all intermediate approximations).
- Compute the set of states where $\phi$ holds with the Emerson-Lei's algorithm (give all intermediate approximations).
- Explain in natural language the meaning of formula $\phi$. 